1. Discrete-Time Signals
    1. Discrete-time Signals and Signal Processing
    2. Plotting Discrete-Time Signals
    3. Signal Properties
    4. Key Discrete-time Test Signals
    5. Real and Complex Sinusoidal Signals
    6. The Peculiarity of Discrete-Time Sinusoids
    7. Complex Exponentials
2. Signals Are Vectors
    1. Signals are Vectors
    2. Linear Combinations of Vectors
    3. The Strength of a Vector
    4. Inner Products and Orthogonality
    5. Norms and Inner Products of Infinite Length Vectors
3. Discrete-Time Systems
    1. Derived copy of Discrete-time Systems
    2. Linear Systems
    3. Time-Invariant Systems
    4. Linear Time-Invariant Systems
4. Convolution
    1. The Impulse Response of Discrete-Time Systems
    2. Convolution of Infinite-Length Discrete-Time Signals
    3. Convolution of Finite-Length Discrete-Time Signals
    4. Properties of Discrete-Time Convolution
    5. Discrete-Time Infinite-Length and Finite-Length Convolution Equivalence
    6. Impulse Response and LTI System Causality
    7. Impulse Response and LTI System Stability
5. Orthogonal Bases and the Discrete Fourier Transform
    1. Orthogonal Bases

    2. Eigenanalysis of LTI Systems (Finite-Length Signals)
    3. The Discrete Fourier Transform
    4. Discrete Fourier Transform Properties
    5. The Fast Fourier Transform
    6. Fast Convolution with the FFT
    7. Other Orthogonal Bases

6. The Discrete-Time Fourier Transform

    1. Eigenanalysis of LTI Systems (Infinite-Length Signals)
    2. The Discrete-Time Fourier Transform
    3. Discrete-Time Fourier Transform Examples
    4. The Discrete-Time Fourier Transform of a Sinusoid
    5. Discrete-Time Fourier Transform Properties

7. The z-Transform

    1. The z-Transform
    2. The z-Transform Region of Convergence
    3. The Transfer Function of Discrete-Time LTI Systems
    4. z-Transform Properties
    5. The Inverse z-Transform

A World of Signals and Signal Processing

Technological innovations have revolutionized the way we view and interact with the world around us. Editing a photo, re-mixing a song, automatically measuring and adjusting chemical concentrations in a tank...each of these tasks requires real-world data to be captured by a computer and then manipulated digitally to extract the salient information. Have you ever wondered how signals from the physical world are sampled, stored, and processed without losing the information required to make predictions and extract meaning from the data? **Signal processing** is the study of signals and systems that extract information from the world around us.

Signals, Defined

Perhaps the best place to start the study of signal processing is with a dictionary definition:

Signal*

> A **signal** is "is a detectable physical quantity...by which messages or information can be transmitted."

*"Signal." Merriam-Webster.com

This information aspect is critical to us. In other words, signals carry information. Signals are all around us; we encounter them throughout our day. Speech signals, for example, transmit language from one person to another via acoustic waves. If you're interested in looking for airplanes or other targets and sensing them by electromagnetic waves, you can use radar signal processing. Electrophysiological signals carry information about processes that are going on inside our bodies; tools like EKGs or MRIs work with those signals. And finally, financial signals transmit information about events in the economy, signals like stock prices over time or other economic markers.

Signals are Functions

A signal is a function that maps an *independent* variable to a *dependent* variable. Mathematically, we're going to think of signals as functions, and a function is simply a mapping from an independent variable that we can change onto a dependent variable that depends on that independent variable. The terminology $x[n]$ is how we're going to denote a signal. It consists of an independent variable, n, that for each of its values produces the another value $x[n]$.

Discrete-Time Signals

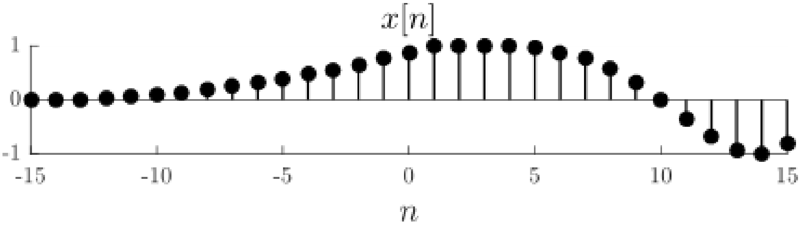
Perhaps you are wondering about the use of brackets--instead of parentheses--in our signal function notation x[n]. This is the typical way to refer to **discrete-time** signals. A discrete-time signal is a signal where the independent variable n is an integer (as opposed to a continuous-time signal x(t), whose independent variable t is a real number).

Plotting Discrete-Time Signals

Recall that a **discrete-time signal** is a **function** with an integer-valued independent variable n. The variable n marches through time from negative infinity to positive infinity. For each value of n, we get the value of our function x[n]. Now, that x[n] is either going to be a **real number**, meaning it's going to live in the real number set, or it's going to be a **complex number** and live in the complex number set.
Example of a discrete-time signal.

Plotting Real Signals

We're going to see a lot graphs in our study of signal processing:

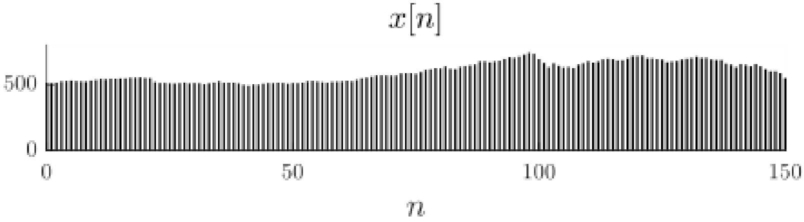


For each value of one of these n, we get the value of x[n]. For clarity, we're often going to color in these circles at the top, but that's really just a matter of taste. We're either going to label the signal on the y-axis or, more typically, in the title of the graph.
A financial series signal.Daily temperatures over the course of a year. The discrete-time plot of a

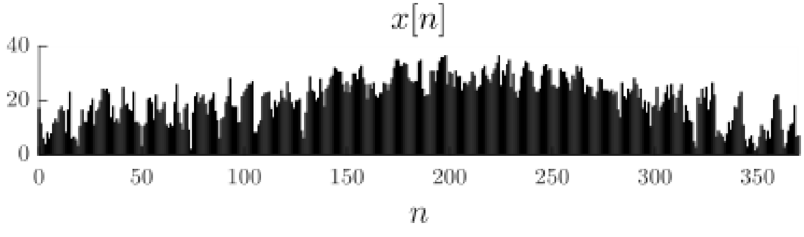
speech signal.

Examples of Discrete-Time Signal Plots

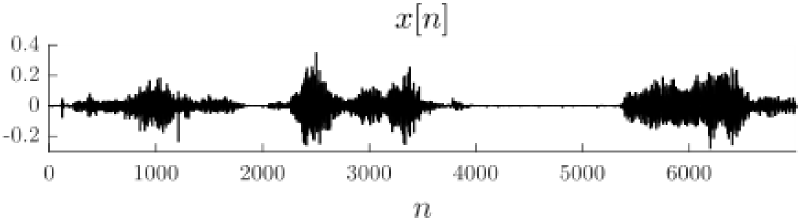
Here are some examples of signals. The first is a financial time series. It's the daily closing share price of Google for a five-month period:



You can see here that it's a discrete time signal, where each of these signal points corresponds to one single share price at the end of a day. There are some fluctuations in the price, and if you were a financial trader or if you were an economist, you would be very interested in the information that this daily closing share price signal conveys. Another example is a temperature signal, the temperature at Houston Intercontinental Airport every day at noon for 365 days that comprise the year 2013 (in degrees Celsius):



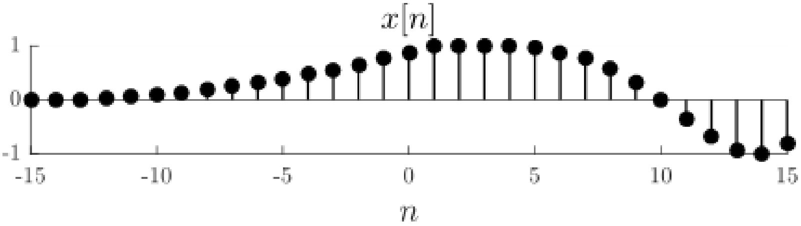
Again, we can see that there are fluctuations in this signal, and if you were a meteorologist or a climatologist, you'd be very interested in the information that this signal conveys. Finally, here's an audio signal that is speech from an actor speaking a part in Shakespeare's play, *Hamlet*:

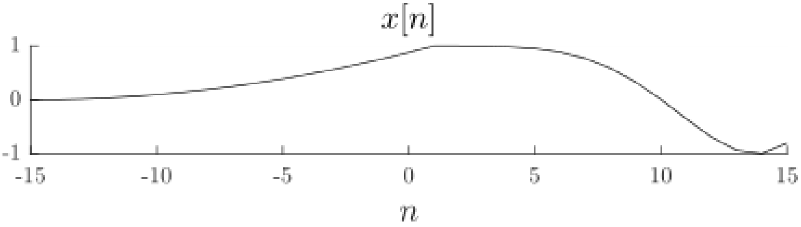


Discrete-time signals are undefined between the integer index values and should be plotted accordingly. The second plot interpolates between the discrete-time integer index values, which is inappropriate for a discrete-time plot.

Plotting Discrete-Time Signals Correctly

We need to remember that with a discrete-time signal, the independent variable is *integer* valued. This means that when you plot a signal in a program like MATLAB, you must use a discrete-time plotting function (like the `stem` function) that respects the fact that the signal is only defined at discrete, integer time points, rather than a function (like `plot`) which interpolates between points:





Plotting Complex-valued Signals

Up to this point, we've been talking about real-valued signals. They comprise a single plot of n versus $x[n]$. But what about complex-valued signals?

Recall that a complex number has a real component and an imaginary component. There are two equivalent ways of expressing a given complex number. For some $a \in C$, we can express a in two different ways:

- Cartesian/rectangular form: $a = \mathrm{Re}(a) + j\ \mathrm{Im}(a)$
- Polar form: $a = |a| e j \angle a$,

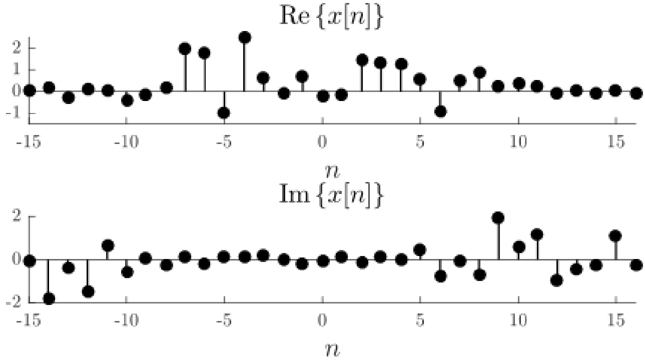
where $j = -1$ (in engineering contexts the variable j is used to represent this value because i represents electrical current). Just as a complex number can be expressed in two different ways, so can a complex-valued signal:

- Cartesian/rectangular form: $x[n] = \mathrm{Re}(x[n]) + j\ \mathrm{Im}(x[n])$

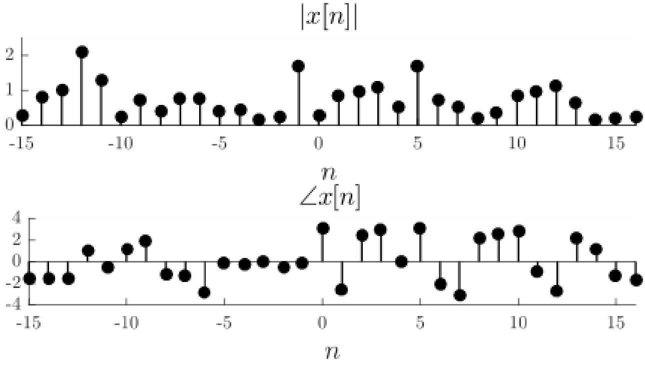
- Polar form: $x[n] = |x[n]|ej\angle x[n]$

What this means is that, if we're plotting a complex-valued signal, we actually need *two* plots. As we have seen, there are two different ways we can plot the same complex-valued signal:

- Cartesian/rectangular form:Plots of the real and imaginary parts of a complex-valued signal x[n].



- Polar form:Plots of the magnitude and phase of a complex-valued signal x[n].



Signal Properties

Signal Classification

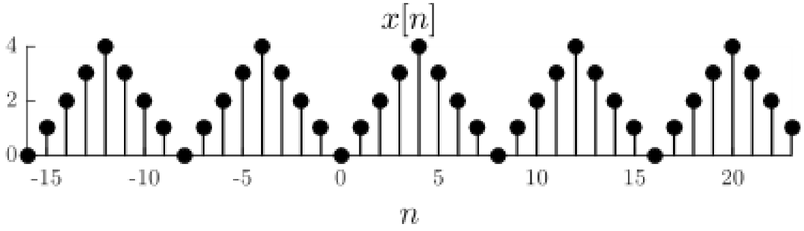
Signals can be broadly classified as discrete-time or continuous-time, depending on whether the independent variable is integer-valued or real-valued. Signals may also be either real-valued or complex-valued. We will now consider some of the other ways we can classify signals.

Signal Length: Finite/Infinite

This classification is just as it sounds. An **infinite-length** discrete-time signal takes values for all time indices: all integer values n on the number line from $-\infty$ all the way up to ∞. A finite-length signal is defined only for a certain range of n, from some N1 to N2. The signal is not defined outside of that range.
A periodic discrete-time signal. Note how it repeats every 8 time units.

Signal Periodicity

As the name suggests, **periodic** signals are those that repeat themselves. Mathematically, this means that there exists some integer value N for which $x[n+N]=x[n]$, for all values of n. So if we define a fundamental period of this particular signal of length, like $N=8$, then we will see the same signal values shifted by 8 time indices, by $16, -8, -16$, etc. Below is an example of a periodic signal:



So periodic signals repeat, and clearly periodic signals are going to be, therefore, infinite in length. It's also important to remember that to be periodic in discrete-time, the period N must be an integer. If there is no such integer-valued N for which $x[n+N]=x[n]$ (for all values of n), then we classify the signal as being **aperiodic**.

An infinite-length signal x[n] (only part of it shown) has a portion extracted via windowing to create a finite-length signal y[n]. An infinite-length signal x[n] (only part of it shown) has a portion extracted via windowing to create a finite-length signal y[n]. A finite-length signal x[n] is periodized to create an infinite-length signal y[n] (only a portion of it is shown).

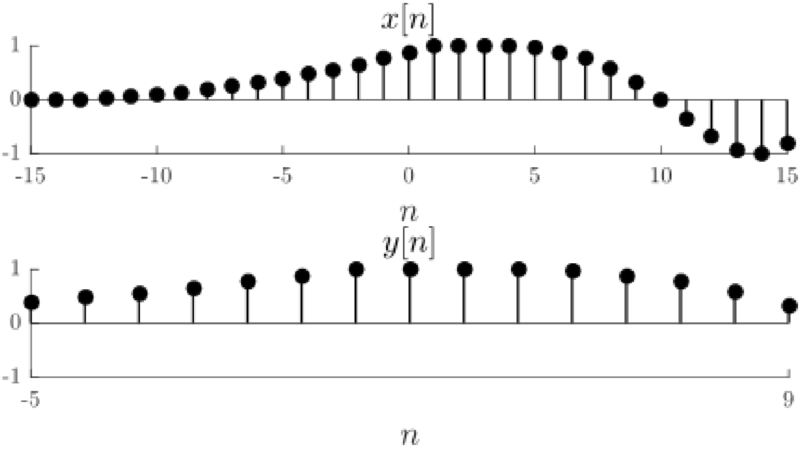
Converting Between Infinite and Finite Length

In different applications, the need will arise to convert a signal from infinite-length to finite-length, and vice versa. There are many ways this operation can be accomplished, but we will consider the most common.

The most straightforward way to create a finite-length signal from an infinite-length one is through the process of **windowing**. A windowing operation extracts a contiguous portion of an infinite-length signal, that portion becoming the new finite-length signal. Sometimes a window will also scale the smaller portion in a particular way. Below is a mathematical expression of windowing (without any scaling):

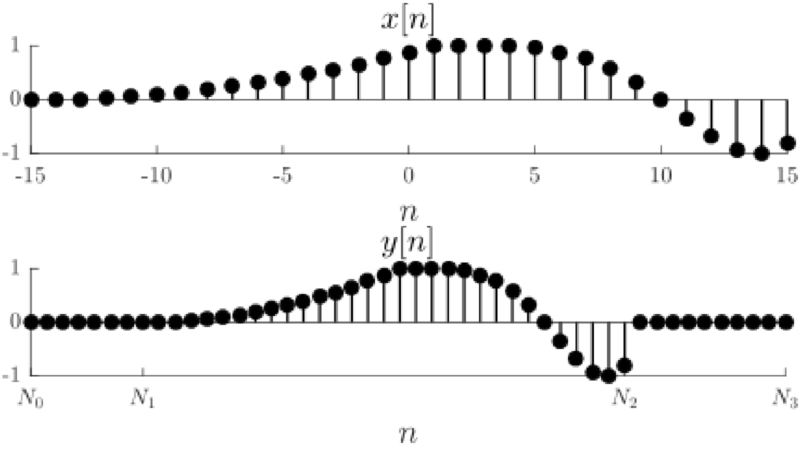
$y [n] = \{ x [n] \; N 1 \leq n \leq N 2 \; \text{undefined else}$

Below is a signal x[n] (assume it is infinite-length, with only a part of it shown), with a portion of it extracted to create y[n]:



There are two ways a signal can be converted from finite-length to infinite-length. The first is referred to as **zero-padding**. It is easy to take a finite-length signal and then make a larger finite-length signal out of it: just extend the time axis. We then have to decide what values to put in the new time locations, and simply putting 0 at all the new locations is a common approach. Here is how it looks, mathematically, to create a longer signal y[n] from a shorter signal x[n] defined only on $N1 \leq n \leq N2$:

$$y[n] = \begin{cases} 0 & N0 \leq n < N1 \\ x[n] & N1 \leq n \leq N2 \\ 0 & N2 < n \leq N3 \end{cases}$$

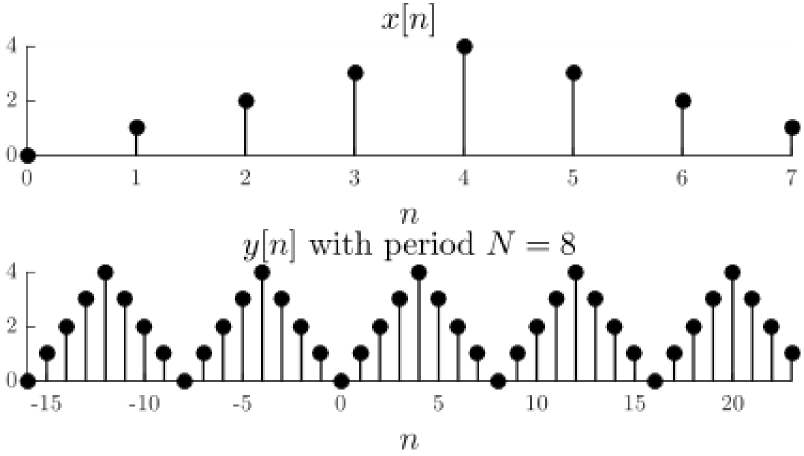


Here, obviously $N0 < N1 < N2 < N3$, and if we extend N0 and N3 to negative and positive infinity, respectively, then y[n] will end up being infinite-length.

The other way to make an infinite-length signal from a finite-length one is to **periodize** it, which means replicating a finite-length signal over and over to create an infinite-length periodic version. Mathematically, that means defining the new infinite-length periodic signal like this:

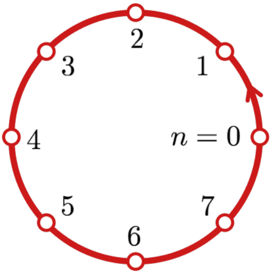
$$y[n] = \sum_{m=-\infty}^{\infty} x[n - mN], N \in Z = \cdots + x[n + 2N] + x[n + N] + x[n] + x[n - N] + x[n - 2N] + \cdots$$

Graphically, we can see that this amounts to repeating the signal over and over, before and after the original portion:

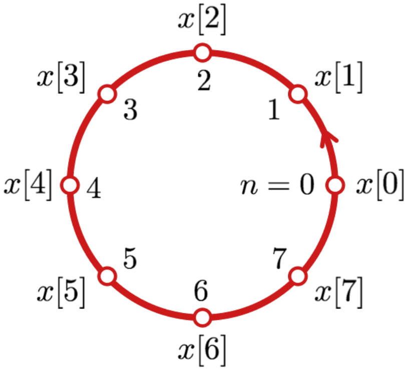


Periodization and Modular Arithmetic

It turns out that, as we consider periodization and periodic signals, the notion of modular arithmetic will be helpful. In modular arithmetic, integers do not lie on a line stretching from negative infinity to infinity, but rather on a circle of a defined size N. In modulo-8, for example, the numbers are 8 spaces on a "wheel." Our convention will be for the numbers to traverse from 0 to $N-1$, counter-clockwise:



Consider a finite-length signal of size N. We can align the time-dependent values of the signal on the modulo circle:



When we travel around the clock once, from time index 0 to 7, we express the finite-length signal. But if we keep traveling, in one direction or the other, then that amounts to periodizing the signal. Using our modulo notation, we can periodize a finite-length signal $x[n]$ to be an infinite-length periodic signal $y[n]$ like this:$y[n]=x[(n)N]$. The modulo operation $(n)N$ returns the integer remainder after n is divided by N. For example, (13)8 is equal to 5, because 8 goes into 13 once, with a remainder of 5. For periodization, this would result in $y[13] = x[(13)8] = x[5]$.

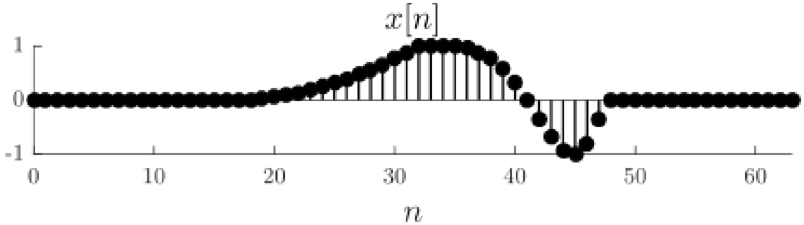
Finite/Periodic Signals Relationship

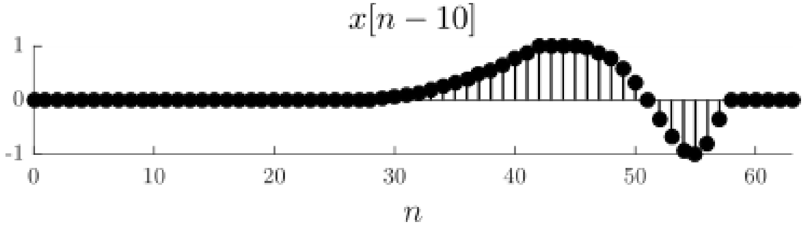
We have seen that we can take an N-length finite-length signal and periodize it to make an infinite-length periodic signal with a period of N. By the same token, we can also work in reverse and extract one period worth of signal from any periodic signal to create a finite-length signal. **What this means is we can consider periodic signals and finite-length signals to be essentially equivalent**: we can consider just one period of a periodic signal (the rest of the signal is redundant, by definition), or periodize a finite-length signal. They are two ways of looking at the same thing, a phenomenon we will often see in our study of signals and systems, and we will choose the perspective that best suits our needs for particular applications.

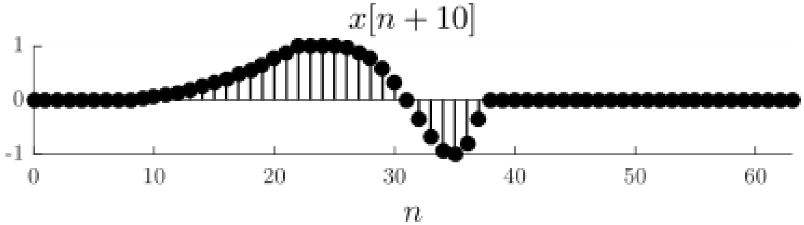
A signal $x[n]$ shifted according to the expression $x[n - m]$ where m is positive, and negative. A periodic $y[n]$ signal shifted one value in time to the right, $y[n - 1]$.

Shifting Infinite-length Signals

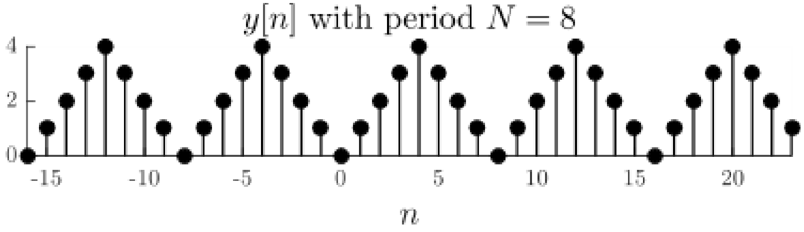
Given some signal $x[n]$, it will often be necessary for us to consider that signal shifted in time. We denote such a shift mathematically with an expression like x[n−m], where m is some integer. If m is greater than zero, then x[n−m] will be just like x[n], except it will be shifted to the right by m time units. If m is less than zero, it will be shifted to the left. Here is who that might look for a couple values of m:

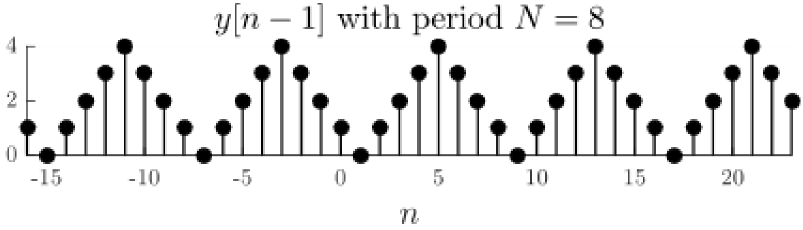






This type of shifting works the same with periodic signals:





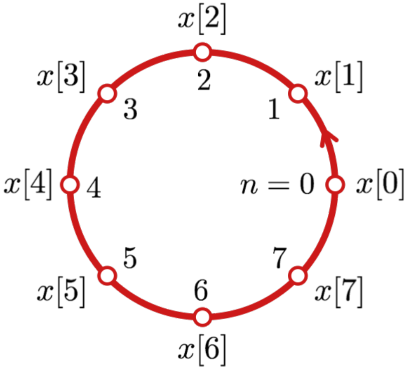
Of course, for periodic signals, certain shifts actually do not have any effect on them. If a signal repeats with a period of N, then shifting that signal by any integer multiple of N will yield the original signal. Take a look at the signal above, which was shifted to the right by a time unit of 1. If we keep on shifting it until it is shifted 8 time units the result will be identical to the original signal.

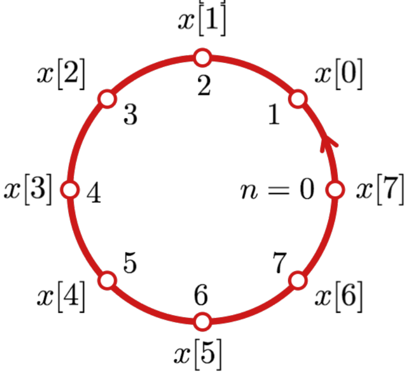
Circularly shifting a signal by m amounts to turning it counter-clockwise m steps. Here is the original signal x[n] and its shifted version x[(n − 1)8].

Shifting Finite-Length Signals

Can finite-length signals be shifted, as well? There does not seem to be any reason why not. Suppose we have some finite-length signal x[n] (of length N) and we define another finite-length signal v[n] to be v[n] = x[n − 1]. So we have v[1] = x[0] and v[2] = x[1] and so on until v[N − 1] = x[N − 2]. But what about v[0], what do we put there? And how about x[N − 1], where is that supposed to go? We do not want to invent information to put in v[0], nor lose the information of x[N − 1]. An elegant solution is to periodize x[n], and then consider the relation v[n] = x[n − 1]. In this case, we now have a value for v[0]: v[0] = x[− 1]. Since x[n] is periodic with period N, it also happens that x[− 1] = x[N − 1], so we do not lose that information.

This kind of operation, for finite-length signals, is called a **circular shift**, and we can express it mathematically with the help of our modular arithmetic operator. Circularly shifting a finite-length signal x[n] by m time units is expressed as x[(n − m)N]. It can also be visualized by turning x[n] about the circle on which it resides:

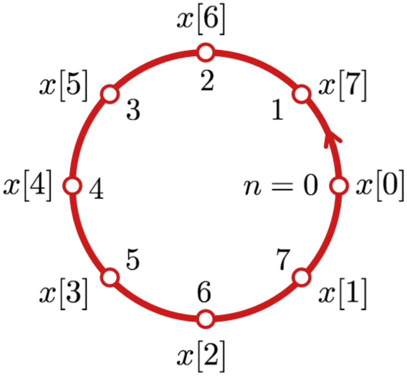




We can time reverse a finite-length signal x[n] with the mathematical expression x[(− n)8]. Note here how the signal values are traversing clockwise, i.e., backwards in time.

Time Reversing Finite-Length Signals

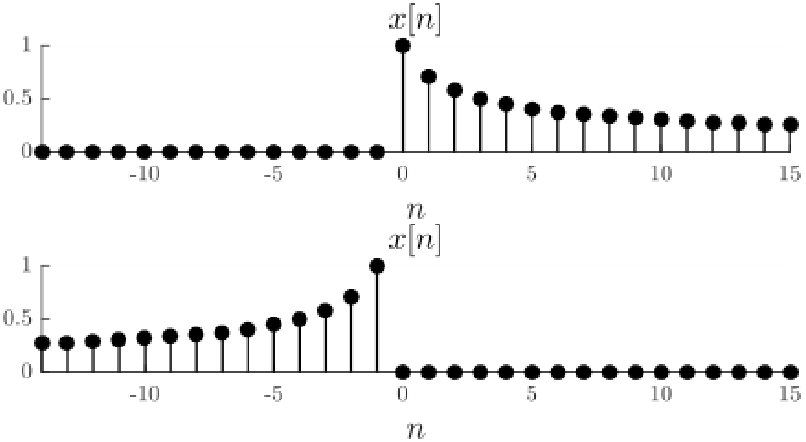
For infinite length signals, the transformation of reversing the time axis $x[-n]$ is obvious: just flip the signal about $n = 0$. But things are not quite so obvious for finite-length signals; if a signal is defined for, say, n between 0 and N, then what gets flipped across the $n = 0$ from the negative side? Once again, it turns out the modular arithmetic operator can be called in for help. We reverse the time axis, modulo N: $x[(-n)N]$. Below is an image of a finite-length ($N = 8$) signal, time-reversed:



A causal signal is 0 for all $n < 0$, whereas an acausal signal is 0 for all $n \geq 0$.

Signal Causality

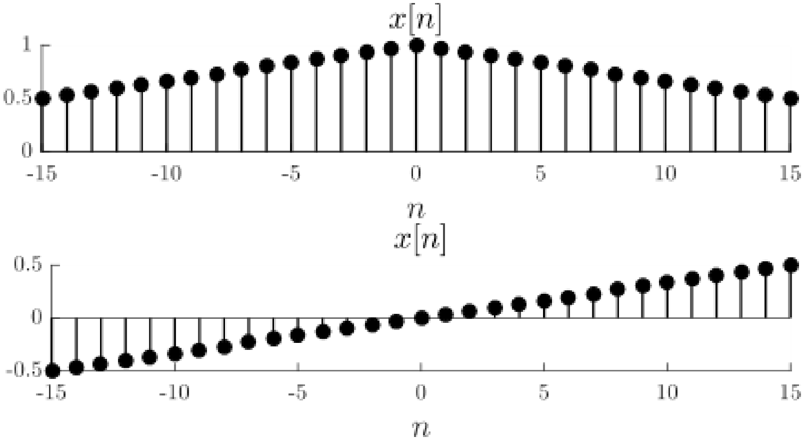
A signal $x[n]$ is **causal** if $x[n] = 0$ for all $n < 0$, it is **anti-causal** if $x[n] = 0$ for all $n \geq 0$, and it is **acausal** if is neither causal nor anti-causal.



For an even signal, $x[-n]=x[n]$. For an odd signal, $x[-n]=-x[n]$.

Even and Odd Signals

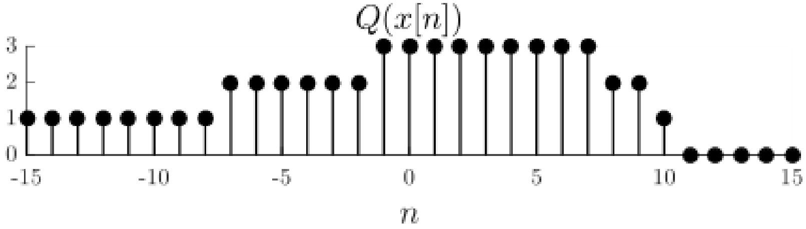
A signal $x[n]$is defined as **even** if $x[-n]=x[n]$, and **odd** if $x[-n]=-x[n]$.



A digital signal with $q=2$ bits, so $D=22=4$ levels.

Digital Signals

Digital signals are a special sub-class of discrete-time signals. While the independent time variable for discrete-time signals is integer-valued, the dependent variable (i.e., the value the signal takes at any given time) can take on any value. However, for digital signals, both of these variables are discrete-valued. Rather than take any value on a continuum, discrete signals take only a limited number of values, or **levels**. Typically, the number of levels is expressed as D = 2q, and each possible value of x[n] is represented as a digital code with q bits.



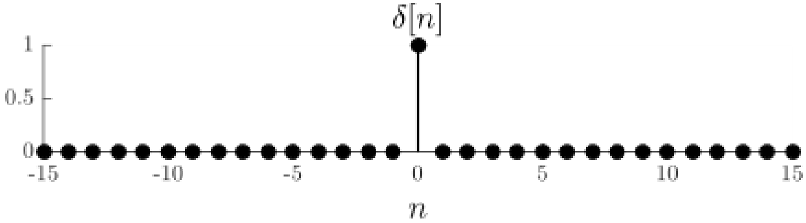
Key Discrete-time Test Signals

In our study of discrete-time signals and signal processing, there are five very important signals that we will use both to illustrate signal processing concepts, and also to probe or test signal processing systems: the **delta function**, the **unit step function**, the **unit pulse function**, the **real exponential function**, **sinusoidal functions**, and **complex exponential functions**. We will initially consider the first four; sinusoids and complex exponentials are particularly important and will be treated separately. Each of these signals will be introduced as infinite-length signals, but they all have straightforward finite-length equivalents.
 The discrete-time delta function. A time-shifted discrete-time delta function $\delta[n-m]$, where $m=9$. Using a time-shifted delta function to isolate a sample of the signal $x[n]$.

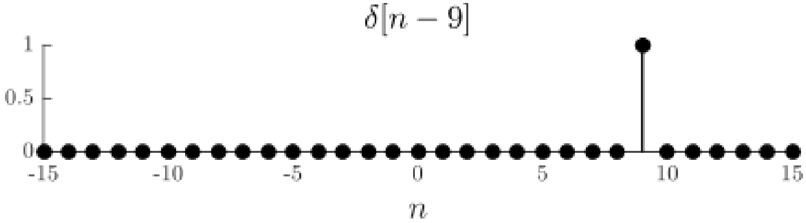
The Discrete-time Delta Function

The **delta function** is probably the simplest nontrivial signal. It is represented mathematically with (no surprise) the Greek letter delta: $\delta[n]$. It takes the value 0 for all time points, except at the time point $n=0$ where it peaks to the value 1:

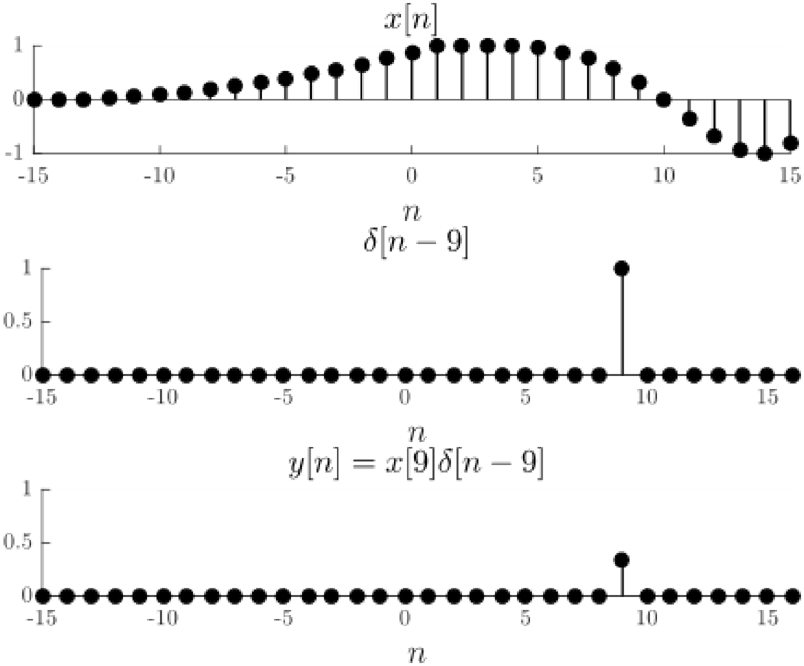
$$\delta [n] = \{ 1 \; n = 0 \; 0 \text{ else}$$



In a variety of important settings, we will often see the delta function shifted by a particular time value. The delta function δ[n − m]is 0, except for a peak of 1 at time m:



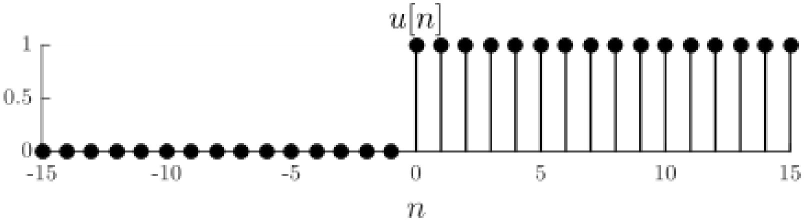
One of the reasons the shifted delta function is so useful is that we can use it to select, or sample, a value of another signal at some defined time value. Suppose we have some signal x[n], and we would like to isolate that signal's value at time m. What we can do is multiply that signal by a shifted delta signal. We can say y[n] = x[n]δ[n − m], but since that y[n] will be zero for all n except at n = m, it is equivalent to express it as y[n] = x[m]δ[n − m], where now x[m] is no longer a function, but a constant. The following figure shows how this operation isolates a particular time sample of x[n]:



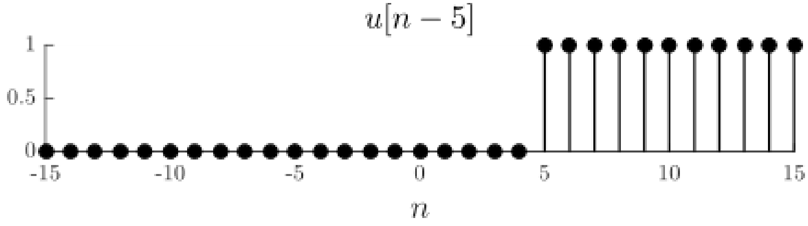
The unit step function. A shifted step function $u[n - m]$ with $m = 5$. A shifted step function can be used to zero out all values of a signal before a certain time index.

The Unit Step Function

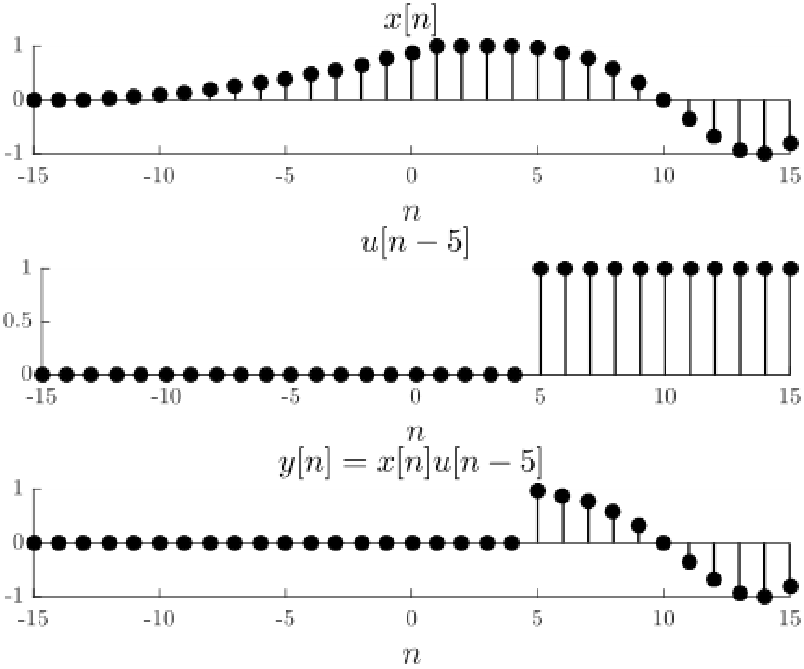
The **unit step function** can be thought of as turning on a switch. Usually identified as $u[n]$, it is 0 for all $n < 0$, and then at $n = 0$ it "switches on" and is 1 for all $n \geq 0$: $u[n] = \{0 n < 0 1 n \geq 0$:



As with the delta function, it will also be useful for us to shift the step function:



And, as you might have guessed, we can use a shifted step function in a similar way to the delta function by multiplying it with another signal. Whereas the delta function selected a single value of a certain signal (zeroing out the rest), the step function isolates a portion of a signal after a given time. Below, a step function is used to zero out all the values of x[n] for n < 5, keeping the rest:

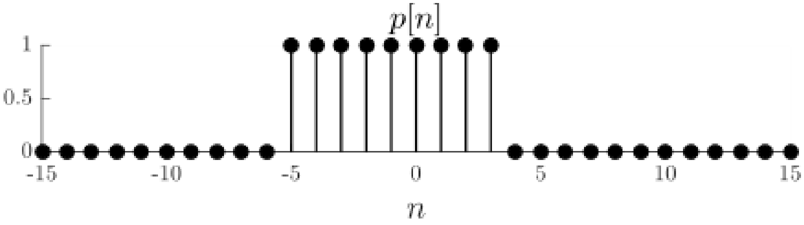


Supposing a signal x[n] were not causal, setting m to zero and performing the operation x[n]u[n] would zero out all values of x[n] before $n = 0$, thereby making the result causal.
The unit pulse function p[n], here with $N1 = -5$ and $N2 = 3$.

The Unit Pulse Function

The **unit pulse function** p[n] is very similar to the unit step function in that it "switches on" from 0 to 1 at a certain time, but then it also "switches off" at a later time. We will say it "switches on" at time N1, and "off" at time N2:

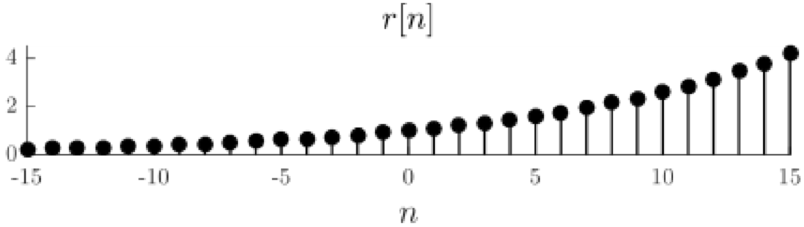
$p\,[\,n\,] = \{\, 0\ n < N\,1\ 1\ N\,1 \le n \le N\,2\ 0\ n > N\,2$

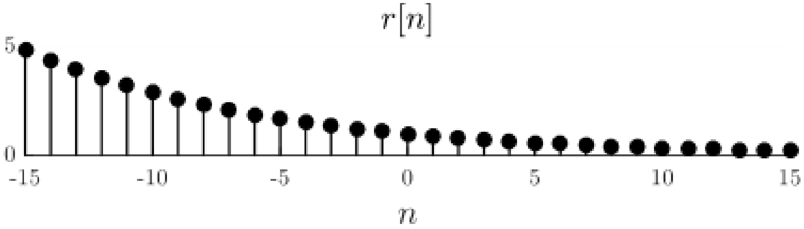


Of course, rather than use the above piece-wise notation, it is also possible to express the pulse as the difference of two step functions: $p[n] = u[n - N1] - u[n - (N2 + 1)]$.
For $a > 1$, the real exponential function increases with time. Here $a = 1.1$. For $0 < a < 1$, the real exponential function decreases with time (or we could say it increases exponentially as the time index decreases). Here $a = .9$.

The Real Exponential Function

Finally, we have the **real exponential function**, which takes a real number a and raises it to the power of n, where n is the time index: r[n] = an, a∈R. So at n = 0, r[n] = a0, at n = 1 it equals a, is a2 at n = 2, and so on. As the name suggests, the signal will exponentially increase or decrease over time, depending on the value of a.





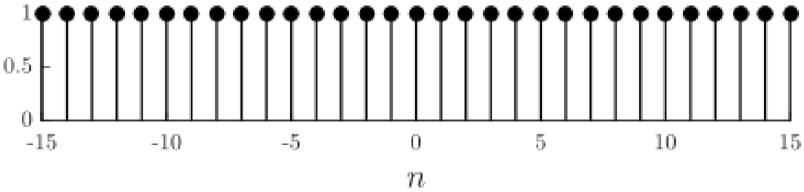
Real and Complex Sinusoidal Signals

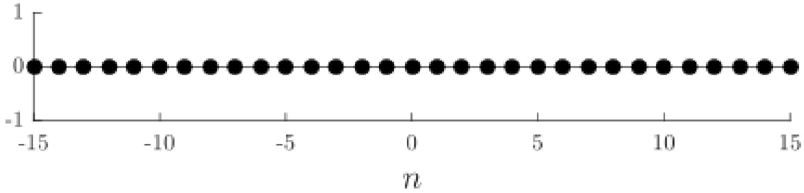
Discrete-time real and complex-valued sinusoidal signals are an incredibly important signal class in the study of discrete-time signals and systems. Sinusoidal waves show up in all sorts of science and engineering applications, but they are particularly relevant for signal processing because (as we will see later) they are the foundation of Fourier analysis.

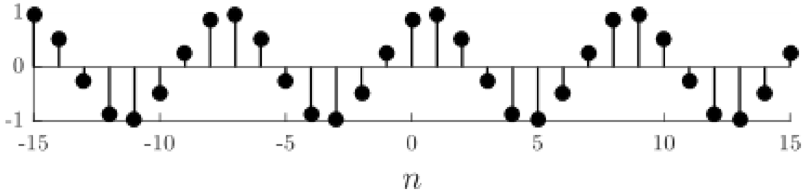
(a) A plot of $\cos(0n)$. At every point in time, this signal takes the value $\cos(0)=1$. (b) A plot of $\sin(0n)$. At every point in time, this signal takes the value $\sin(0)=0$. (c) A plot of $\sin(\pi 4n+2\pi 6)$. (d) A plot of $\cos(\pi n)$. Note how when $\omega=\pi$, as in this example, the signal is oscillating as rapidly as possible, between -1 and 1 at every single time instance. This phenomenon is the opposite of when $\omega=0$, for which the signal does not oscillate at all. So in some sense we can see that 0 is the lowest possible frequency ω, and π is the highest. Four cosine waves with the same frequency, but different phases: $\cos(\pi 6n-0)$, $\cos(\pi 6n-\pi 4)$, $\cos(\pi 6n-\pi 2)=\sin(\pi 6n)$, and $\cos(\pi 6n-2\pi)=\cos(\pi 6n)$. Note how a phase shift of $\pi 2$ shifts the cosine to be a sine wave, and a phase shift of 2π shifts it all the way over to where it was before the phase shift.

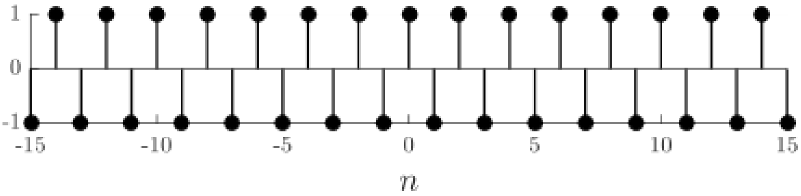
Real Valued Sinusoids

There are two real-valued discrete-time sinusoidal wave signals: the **sine wave** signal and the **cosine wave** signal. They are represented mathematically as $\sin(\omega n + \phi)$ and $\cos(\omega n + \phi)$. Let's take a look at those in more detail. First, as we have seen with other discrete-time signals, n is the independent variable time index, and it runs from negative infinity to infinity. The variable ω is known as the **frequency** of the sinusoidal signal, and we will see how changing the value of ω impacts the rate of the signal's oscillation. The variable ϕ is the **phase** of the signal, and changing it will shift the signal left or right along the time axis. Finally, the terms sin or cos return the corresponding trigonometric value of $\omega n + \phi$ for each value of the time index n. Here are a few examples of real sinusoidal waves:

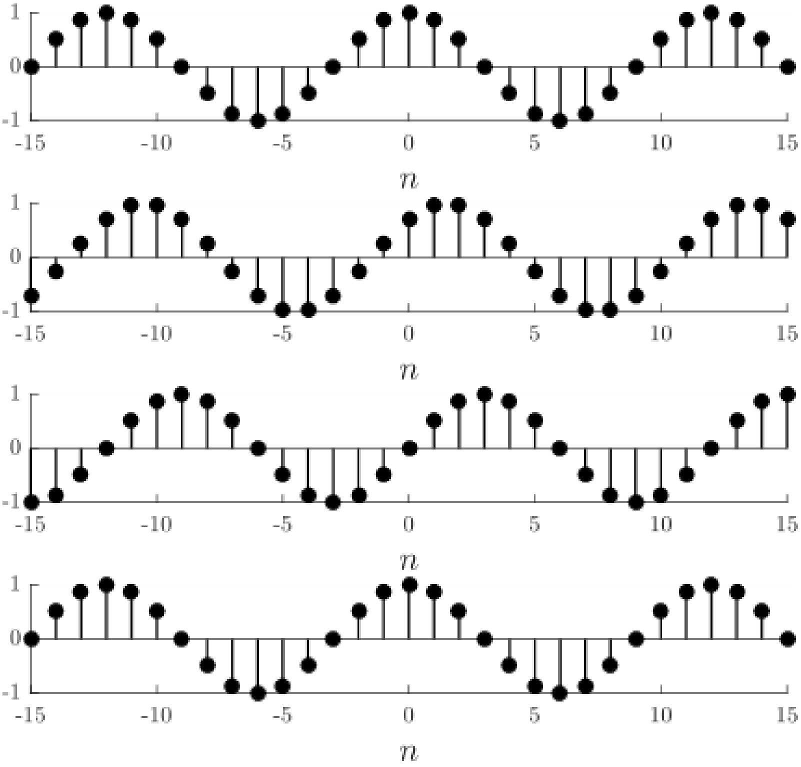








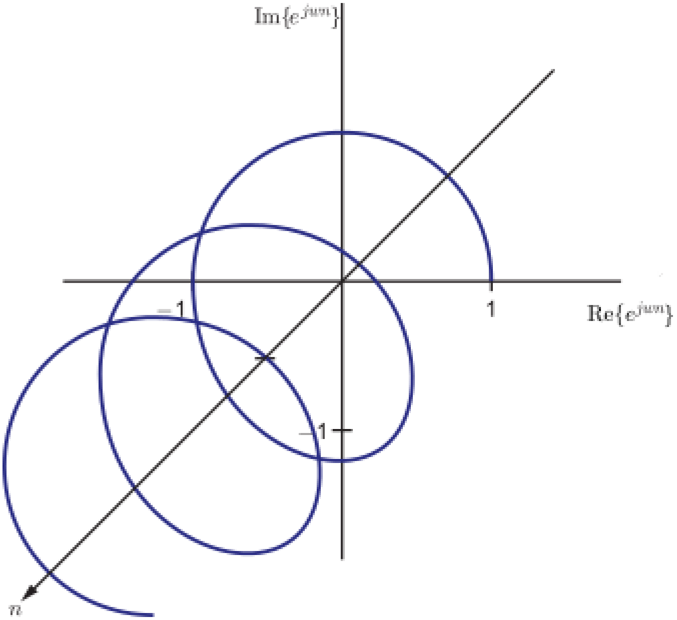
We saw in the figure above how the frequency ω influences the rate of the wave's oscillation. The other variable in the signal, the phase ϕ, can shift the wave backwards and forwards along the time axis, without affecting the frequency. Below are plots of a cosine wave which all have the same frequency, but with a variety of phase shifts (i.e., different values of ϕ):



A three dimensional visualization of a complex sinusoid. Note that this image is continuous-valued, whereas a discrete valued version would actually appear as points along the line. A complex sinusoid plotted according to its magnitude and phase. Note that the magnitude of a single complex sinusoid is trivial, as $|ej(\omega n+\phi)|=1$. A complex sinusoid plotted according to its real and imaginary parts. These are a cosine, and sine, respectively, which follows from Euler's Formula.

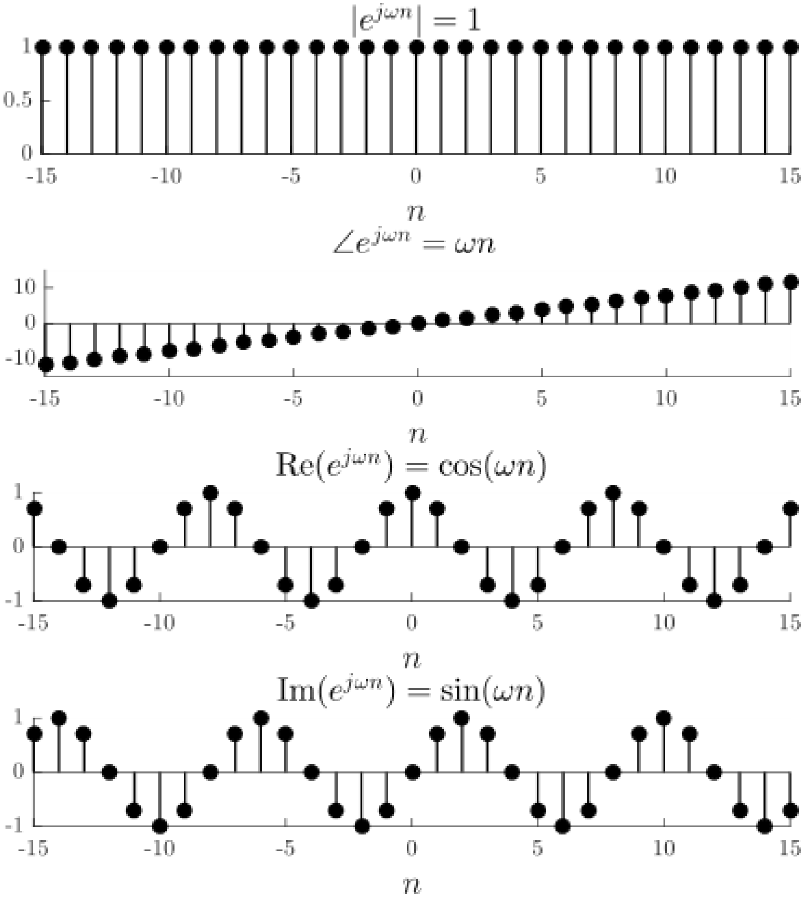
Complex Valued Sinusoids

So we have reviewed the real sine waves sin and cos, and perhaps seeing them in proximity brought to mind a very special relationship called **Euler's Formula**: $ej\theta=\cos(\theta)+j\sin(\theta)$ (you may remember this from math class with an i instead, but recall engineers use that letter for current, and we call the imaginary number j). That formula works for any particular value of θ, so of course it applies when we consider $\omega n+\phi$, as above, which gives us a complex valued sinusoid: $ej(\omega n+\phi)=\cos(\omega n+\phi)+j\sin(\omega n+\phi)$. Let's look at some plots of complex sinusoids. Unlike two-dimensional real sinusoids (which have a one-dimensional independent time variable n and a take a one-dimensional value at each time value), complex sinusoids are three dimensional: they have the time dimension, a real dimension, and an imaginary dimension. So they can be visualized as a three dimensional helix in space:



If you were to look at this helix from directly above, you would see only the real portion of the helix, and it would appear to be a cosine wave. If you looked at it from the side, you would see the imaginary aspect of it, as a sine wave. The frequency variable ω controls how quickly the helix rotates across time n, and also the direction: positive values cause it to rotate in the counterclockwise manner shown, and negative values would result in it rotating clockwise.

While it is illuminating to visualize complex simusiods in three dimensions, in practice it is actually most common to view them in two, separately plotting either the real and the imaginary parts with respect to time, or the magnitude and phase across time:



We'll wrap up our introduction of sinusoids by briefly considering the concept of negative-valued frequencies. It is easiest to see the difference a negative frequency makes, compared to a positive frequency of the same magnitude, by expressing it all mathematically: $e^{j(-\omega)n} = e^{-j\omega n} = \cos(-\omega n) + j\sin(-\omega n) = \cos(\omega n) - j\sin(\omega n)$. So negating the frequency of a complex sinusoid has no effect on the real part of the signal (the cosine), but it flips the sign of the imaginary part (the sine).

This operation (preserving the real part, but changing the sign of the imaginary part) is also known as taking the **complex conjugate** of the signal. So negating the frequency of a complex sinusoid is the same thing as taking the complex conjugate of it:$e^{j(-\omega)n} = e^{-j\omega n} = (e^{j\omega n})^*$.

Why use imaginary numbers?

Now perhaps you are wondering the point of using imaginary numbers. After all, aren't all real world signals, well, real-valued? They are indeed, but we can simply consider them as the real-part of a complex-valued signal. And why go to that trouble? There are many good reasons, but here is one to start with: exponential functions are much easier to work with than trigonometric functions. You can easily simplify $e^a e^b$ into a single term, but you very likely would be turning to a table to simplify $\sin(a)\cos(b)$, wouldn't you?

The Peculiarity of Discrete-Time Sinusoids

Compared to their continuous-time counterparts (those that take a continuous-valued independent time variable t), discrete-time sinusoidal signals have two unique characteristics. It is possible for them to **alias**, and they are not always **periodic**. Here $x1[n]=\cos(\pi 6n)$ and $x2[n]=\cos(13\pi 6n)$. While they have different frequencies, they are identical signals, for $\cos(13\pi 6n)=\cos((\pi 6+2\pi)n)$. Low frequencies are those ω close to 0 or 2π rad, as in the first signal, $\cos(\pi 10n)$. High frequencies are those ω close to π or $-\pi$ rad, as in the second signal, $\cos(9\pi 10n)$.

Aliasing of Discrete-time Sinusoids

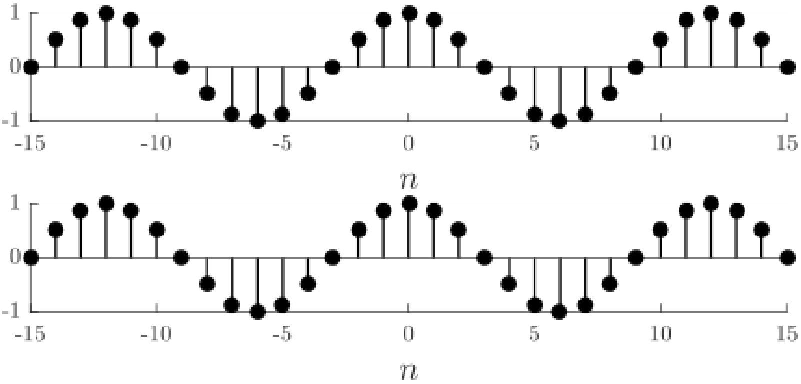
One might think that if two different discrete-time sinusoids had different frequencies, then they would be different signals. Such is always the case with continuous-time sinusoids, but *not* always for the discrete-time version. Consider two discrete-time sinuoids $x1[n]$ and $x2[n]$ with different frequencies, ω and $\omega+2\pi$:

$$x1[n] = ej(\omega n + \phi)$$

$$x2[n] = ej((\omega + 2\pi)n + \phi)$$

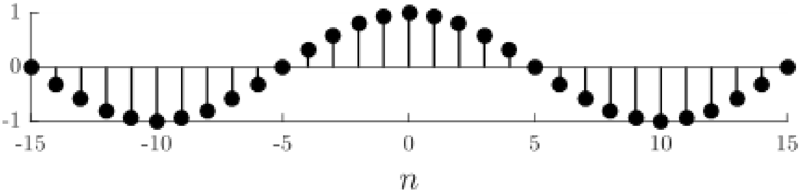
We can then simplify the expression of x2[n], using the fact that $ej2\pi = 1$ to arrive at this surprising conclusion: $x2[n] = ej((\omega + 2\pi)n + \phi = ej(\omega n + 2\pi n + \phi) = ej(\omega n + \phi)ej2\pi n = ej(\omega n + \phi)(1)n = ej(\omega n + \phi) = x1[n]$

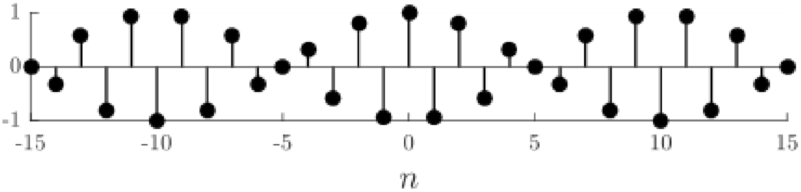
So $x1[n]$ and $x2[n]$ have different frequencies, yet they are identical! You can see this plotted out with $\omega = \pi 6$ below:



This phenomenon is called **aliasing**. It happens when frequencies are offset by any integer multiple of 2π (you can use ω and $\omega + 2\pi k$ in the example above to see for yourself).

So only frequencies along a continuous interval of length 2π on the real number are distinct from each other. For this reason, when we deal with discrete-time frequencies we consider only those along the interval $0 \leq \omega < 2\pi$ or $-\pi < \omega \leq \pi$, as any other frequency aliases back to an identical signal with a frequency in that range. Within these ranges, frequencies close to 0 (or 2π, depending on the range used) are low frequencies--their sinusoids do not oscillate very quickly-- and frequencies close to π (or $-\pi$, depending on the range) are high frequencies:





$x1[n] = \cos(2\pi 316n)$ is periodic, with $N = 16$. $x2[n] = \cos(1.16n)$. The frequency of 1.16 is not a fraction of 2π, so this sinusoidal signal is not periodic.

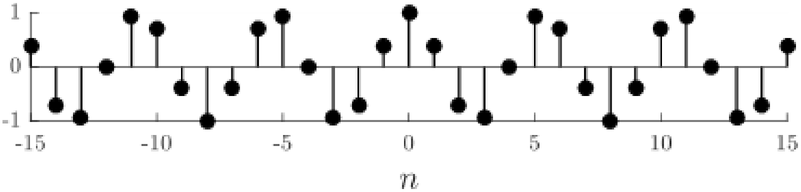
Periodicity of Discrete-time Sinusoids

Recall that a signal $x[n]$ is defined to be periodic if there exists some integer N for which $x[n + N] = x[n]$, $\forall n$. Continuous-time sinusoids are always periodic, but again, such is only sometimes the case for discrete-time sinusoids. Suppose we have a complex sinusoid with a frequency $\omega = 2\pi kN$, where k and N are integers. This just means that ω is some fraction of 2π. It turns out that this signal *is* periodic, with period N:

$x[n] = ej(2\pi kNn + \phi)x[n + N] = ej(2\pi kN(n + N) + \phi) = ej(2\pi kNn + 2\pi kNN + \phi) = ej(2\pi kNn$

$+\phi)ej(2\pi kNN)=ej(2\pi kNn$
$+\phi)ej(2\pi kNN)=ej(2\pi kNn+\phi)(ej(2\pi k))=x[n]$.

Here is a plot of a sinusoid with frequency $2\pi 316$. You will note that it has a period of $N=16$:

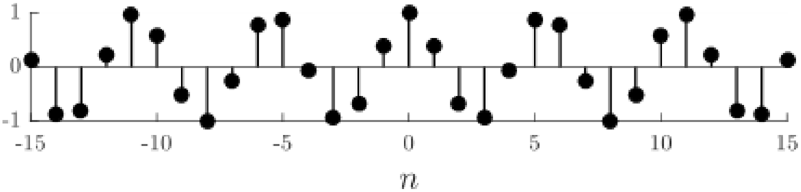


Now, these fractions of 2π are special values of ω we call **harmonic frequencies**, for sinusoids with such frequencies are periodic.

In contrast, consider sinusoids whose frequencies are *not* fractions of π:

$x[n]=ej(\omega n+\phi)x[n+N]=ej(\omega(n+N)+\phi)=ej(\omega n+\omega N+\phi)=ej(\omega n+\phi)ej(\omega N)=ej(\omega n+\phi)ej(\omega N)\neq x[n]$, unless $\omega N=2\pi k\rightarrow\omega=2\pi kN$

So we see that discrete-time sinusoids are periodic if, and only if, their frequencies are fractions of 2π. Consider the example of a non-periodic sinusoid below. It definitely oscillates, and at first it appears to be periodic, but look carefully and you will see that it actually is not, unlike the one from the figure above. Not only is this cosine not periodic, but aside from at respective negated indices (e.g., -1 and 1, -2 and 2), it will never even take the same *value* more than once!



Take note of sinusoids whose frequencies are of the form ω = 2πkN, for they will play a starring role in the Fourier analysis of periodic and finite-length discrete-time signals.

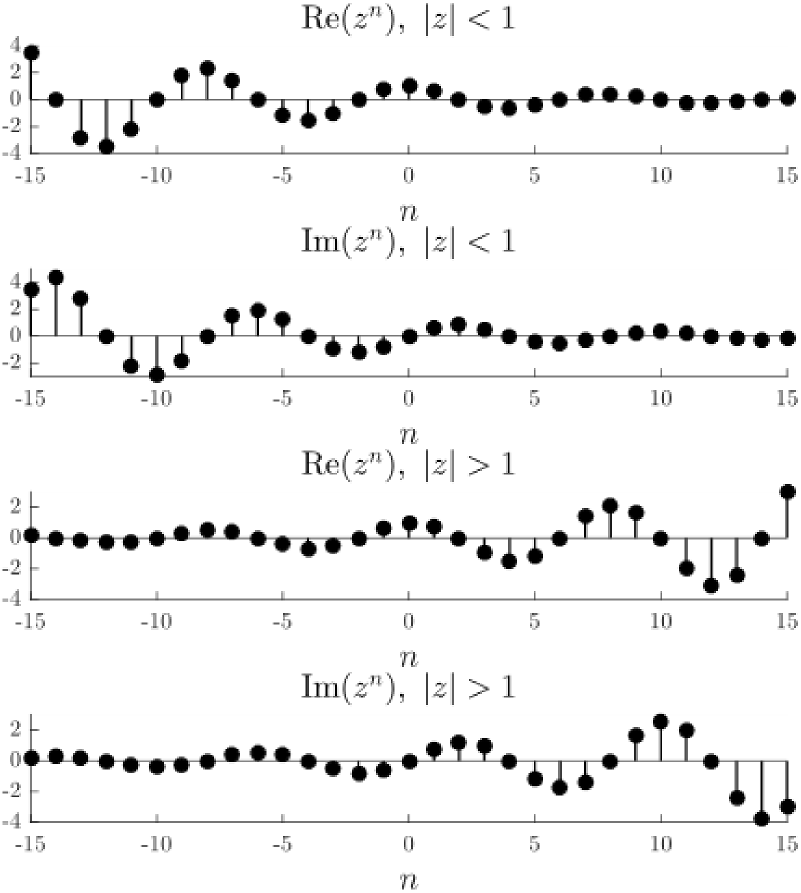
From Complex Sinusoids to Complex Exponentials

Recall the form of a discrete-time complex sinusoid: $x[n] = ej(\omega n + \phi)$. As we have already seen, that signal itself is complex-valued, i.e., it has both a real and an imaginary part. But look closely at just the exponent, and you will see that the exponent itself is purely imaginary.

Suppose we let the *exponent* be complex-valued, say of the form $a + jb$, where a and b are real. Then we have $e(a + jb)n = eanejbn = (ea)nejbn$. The result is a complex sinusoid multiplied by a real exponential signal (whose base is ea).
The real and imaginary parts of a complex exponential xn for which $|z| < 1$. The real and imaginary parts of a complex exponential xn for which $|z| > 1$.

Complex Exponentials, Defined

We do not typically represent complex exponentials in the way derived above, but rather express them in the form $x[n] = zn$, where z is a complex number. Being a complex number, it lies on the complex plane with a magnitude of |z|, and an angle of ∠z we define as ω. So then, if we would like to express x[n] = zn as a combination of a real exponential and a complex sinusoid, as above, we have: x[n] = zn = |z|nejωn. Below are some plots of complex exponentials for different values of z.



We see that when the magnitude |z| is greater than 1, the signal oscillates and exponentially grows with time, and if the magnitude is less than 1, it decays over time. If the magnitude is exactly equal to 1, it does not grow or decay, but only oscillates. In fact, if the magnitude is 1, the complex exponential is, by definition, simply a complex sinusoid: $|1| nej\omega n = ej\omega n$. Therefore you can see that complex sinusoids are a subset of the more general complex exponential signals. They correspond to values of z on the complex plane that lie on the unit circle, $|z| = 1$.

Signals are Vectors

One mathematical way of understanding signals is to see them as functions. A signal $x[n]$ carries some kind of information, having a value $x[n]$ at every given point of its independent time variable n. Another, and complementary, way of understanding signals is to consider them as vectors within vector spaces. By doing this we will be able to apply various tools of linear algebra to help us better understand signals and the systems that modify them.

Vector Spaces

A **vector space** V is a collection of vectors such that if $x,y \in V$ and α is a scalar then $\alpha x \in V$ and $x+y \in V$ (along with some other properties we would expect, such as commutativity, additive and multiplicative identities, etc.). In words, this means that if two vectors are elements of a vector space, any combination or scaled version of them is also in the space. When we consider the scaling factors, we mean α that are either real or complex numbers.

There are many different kinds of vector spaces, but the two in which we are especially interested are RN and CN. RN is the set of all vectors of dimension N, in which every entry of the vector is a real number, and CN is exactly the same, except the entries are complex valued.

Starting Small – A Two Dimensional Vector Space

You are already familiar with a prominent example of a vector space, the two-dimensional real coordinate space R2. Every ordered combination of two real numbers is a vector in this space, and can be visualized as a point or arrow in the two-dimensional Cartesian plane. Suppose x and y are each vectors in R2. $x=[x[0]x[1]]$, $y=[y[0]y[1]]$ where $x[0],x[1],y[0],y[1]\in R$. The indices we use to refer to elements of the vector start their numbering at 0. This is the common convention in signal processing and many programming languages, like C, but note that vector indices in MATLAB start with 1. Scaled versions of these vectors are still within the space:

$$\alpha x = \alpha [x [0] x [1]] = [\alpha x [0] \alpha x [1]] \in R 2$$

So is the sum of two vectors: $x+y=[x[0]x[1]]+[y[0]y[1]]=[x[0]+y[0]x[1]+y[1]]\in R$ When a vector x in RN is scaled by $\alpha\in R$, the result is still in RN. Here is an example of an N-dimensional signal/vector x, which is then scaled by $\alpha=3$. The sum of two vectors in RN is another vector in RN.

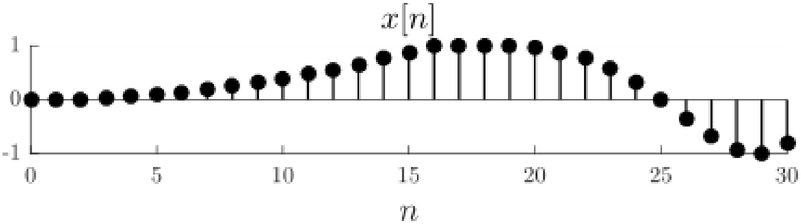
The Vector Space RN

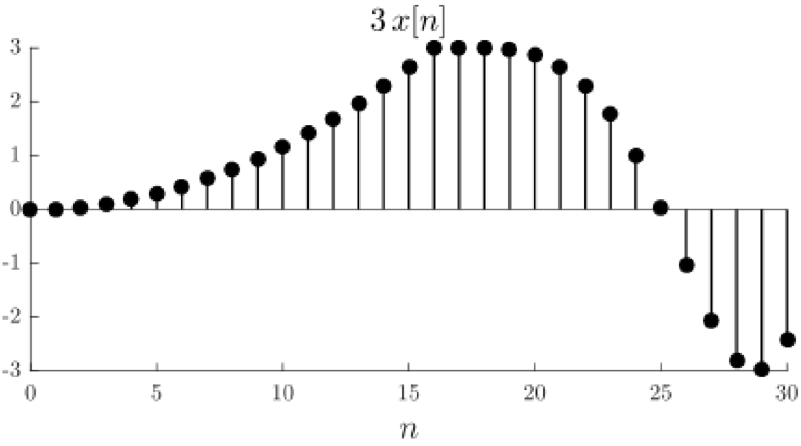
Let's now move from two dimensional vectors to those with N dimensions, each taking a real valued number:

$$x=[x[0]x[1]\vdots x[N-1]],$$

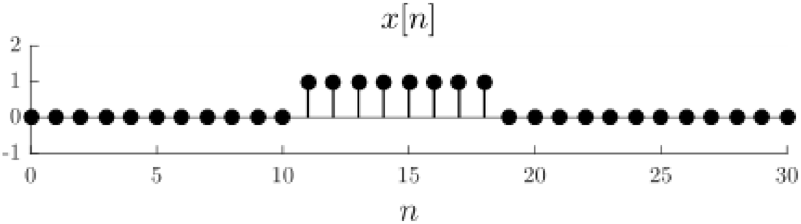
where $x[n]\in R$ So we have a mathematical entity x, with N ordered real values associated with it. Stated this way, we see that x is a signal, simply expressed in a vector form x as opposed to the signal notation form x[n], but both forms refer to the exact same thing.

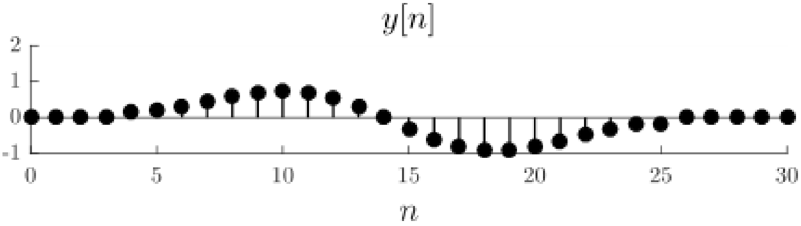
Just as with in 2-dimensions, we can perform the same operations on the N-dimensional signal/vector x:

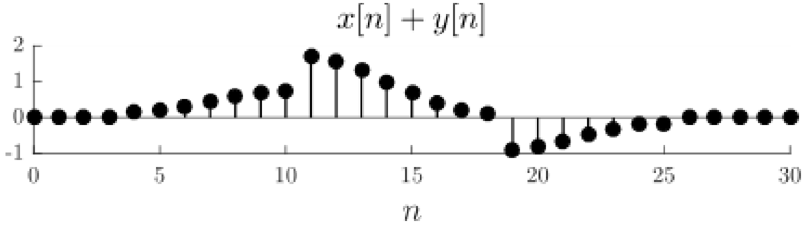




When each of the real-valued elements of x is scaled by a real value α, the results are still real-valued, of course, so the resulting scaled vector is obviously still within RN. As RN is a vector space, that should come as no surprise, and neither should the fact that the sum of two vectors in RN is itself another vector in the vector space RN:







The Vector Space The Vector Space CN

Ordered sequences of N complex numbers form the vector space CN, just like their real counterparts do in RN: $x=[x[0]x[1]\vdots x[N-1]]$,

where $x[n]\in C$. So each element of a vector in CN is a complex number, which can be represented either in terms of its real and imaginary parts, or in terms of its magnitude and phase. The vector as a whole can also be represented in those ways, either in rectangular form

$$x=[\mathrm{Re}\ x[0]+j\ \mathrm{Im}\ x[0]\,\mathrm{Re}\ x[1]+j\ \mathrm{Im}\ x[1]\vdots\mathrm{Re}\ x[N-1]+j\ \mathrm{Im}\ x[N-1]]=\mathrm{Re}\{[x[0]x[1]\vdots x[N-1]]\}+j\mathrm{Im}\{[x[0]x[1]\vdots x[N-1]]\}$$

or in polar form

$$x=[|x[0]|ej\angle x[0]|x[1]|ej\angle x[1]\vdots|x[N-1]|ej\angle x[N-1]].$$

Linear Combinations of Vectors

In the study of signals and signal processing, there is a particular mathematical operation that will show up quite a few times, that of a **linear combination**. Given a collection of vectors in a vector space, say M vectors $x0,x1,\ldots xM-1 \in CN$, and M scalars $\alpha0,\alpha1,\ldots,\alpha M-1 \in C$, then the linear combination of these vectors is:

$$y = \alpha0x0 + \alpha1x1 + \cdots + \alpha M-1xM-1 = \Sigma m = 0M-1\alpha mxm.$$

The result is itself also a vector in the vector space (indeed, this is one of the requirements of what it means for something to be a vector space).

Linear Combination Example: A Mixing Board

A linear combination is a scaled sum of different vectors in a vector space. A real world example of a linear combination is the mixing board used in music recording studios. The board takes in a variety of different inputs and combines them--amplifying some, reducing the level of others--to create a single output of music. Mathematically, we could say x0 is drums, x1 is bass, x2 is guitar, ..., x22 is a saxophone, and x23 a singer, and then the mixing board creates the linear combination $y = \alpha_0 x_0 + \alpha_1 x_1 + \cdots + \alpha_{M-1} x_{M-1} = \sum_{m=0}^{M-1} \alpha_m x_m$. Changing the different α_m's would result in a different kind of sound that either emphasizes or de-emphasizes certain instruments. For example, the producer may be particularly interested in increasing the volume of the cowbell.

Linear Combinations as Matrix Multiplication

It is possible to express a linear combination without explicitly using sums, but rather as the product of a matrix and a vector. To do this, all of the vectors to be scaled and summed in the linear combination must first be arranged into a matrix:

$$X = [x_0 | x_1 | \cdots | x_{M-1}] = \begin{bmatrix} x_0[0] & x_1[0] & \cdots & x_{M-1}[0] \\ x_0[1] & x_1[1] & \cdots & x_{M-1}[1] \\ \vdots & \vdots & \vdots & \vdots \\ x_0[N-1] & x_1[N-1] & \cdots & x_{M-1}[N-1] \end{bmatrix}$$

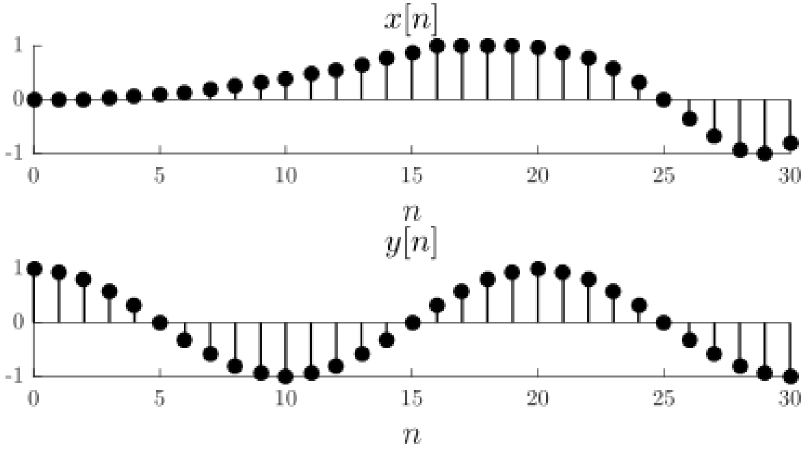
Then the scaling factors for each vector are stacked into a single vector:

$$a = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{M-1} \end{bmatrix}$$

Once those are in place, the linear combination is ultimately expressed as a simple matrix-vector multiplication: $y = \alpha_0 x_0 + \alpha_1 x_1 + \cdots + \alpha_{M-1} x_{M-1} = \sum_{m=0}^{M-1} \alpha_m x_m = [x_0 | x_1 | \cdots | x_{M-1}] \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{M-1} \end{bmatrix} = Xa$.

The Strength of a Vector

In the study of signal processing, the "strength" of signals (which, remember, can be expressed as vectors) is often of interest. It is not immediately clear what standard could be used for judging the strength of a vector. A similar problem comes up when determining the physical strength of a person: does it have to do with the amount of weight that can be lifted, or the number of times a certain weight can be lifted, or perhaps how far the weight can be thrown? So it is with vectors; consider the two below:
Which signal is "stronger"?



Which is stronger than the other? Perhaps you might have intuition just by looking at them, but undoubtedly it will be better if we can introduce some kind of rigorous definition of strength.

Norms: The Strength of Vectors

There indeed are such ways to measure the strength of vectors, and they are called **norms**. Just as there are different ways to measure human strength, there are also different ways to measure the strength of the vector. The first one we will consider is the 2-norm.

The **Euclidean length**, or **2-norm**, of a vector $x \in CN$ is defined as:

$$\| x \| 2 = \Sigma n = 0 N - 1 | x [n] | 2$$

This norm, like all norms, is a function that takes a vector as its argument and then produces a real number that is always greater than or equal to 0. The 2-norm is a very common norm, so often it will be seen without even having a subscript, i.e. as $\|x\|$.

Let's calculate a 2-norm. Suppose:

$$x = [1 2 3]$$

Then $\|x\|2$, the 2-norm of x, is:

$$\|x\|2 = \Sigma n = 0N - 1 | x[n] | 2 = 12 + 22 + 32 = 14.$$

It happens that the 2-norm is actually just one kind of a more general type of norm, called **p-norms**. For any real number $p \geq 1$, the p-norm is:

$$\| x \|_p = \left(\sum_{n=0}^{N-1} | x[n] |^p \right)^{1/p}$$

So the 2-norm is a p-norm with $p = 2$. Another norm of interest to us is when $p = 1$, the **1-norm**:

$$\| x \|_1 = \sum_{n=0}^{N-1} | x[n] |$$

A final norm that we will consider is called the ∞**-norm**. It is essentially what happens as the p in the p-norm increases to ∞, but has this easy to calculate expression: $\|x\|_\infty = \max_n |x[n]|$. So if you have a vector x and take the absolute value of all the entries, the ∞ norm of x is simply the largest of those values.

The Meaning of Different Norms

There is a reason for having different norms; they each are helpful in their own way for measuring particular things about a signal. Some of these measurements have clear physical quantities associated with them. For example, consider the electrical signal travelling through a loudspeaker, and suppose x corresponds to the voltage of this signal. The 2-norm of this signal is significant, because it corresponds to the electrical **energy** going through the coil (although this is actually a bit of a simplification). The ∞-norm of the signal is important because it represents the maximum voltage value going through the circuit. We might want to pay attention to each of those values, though for different reasons. If the 2-norm is too large, then the excessive amount of energy might melt the circuit (or lead to a large electric bill!). If the ∞-norm is too large, the extreme voltage at a given time might cause the speaker cone to move too far, breaking it.

Or consider another example of the utility of different norms. Suppose you are designing an autopilot system for a car. Among other concerns, you clearly will want to make sure it does not unnecessarily deviate from its lane. Call the deviation at a given time the error signal d. Would you seek to minimize $\|d\|2$ or $\|d\|\infty$? If you minimize $\|d\|2$, then the car will, overall, keep to its lane very closely, but quick and large deviations would be permitted: if over the course of a long drive the car stayed in its lane perfectly, except for a fraction of a second it accidentally swerved into another lane, the $\|d\|2$ would be small, but that could potentially still be a disaster! Instead, you would want to minimize the $\|d\|\infty$ value, so the car never moves more than a maximum distance from, say, the center of the lane.

Normalizing a Vector

Suppose you run a recording studio and you are putting different recorded tracks onto a single album. You run in to the following problem: the artist was standing closer to the microphone on some tracks, and farther away on others. The loudness varies wildly across the different tracks, so it would be frustrating for customers to listen to the album, having to change the volume with every track. What would you do?

If you have studied signal processing, then you will know that you can consider each track as a signal, and that each track will have a different norm. In order to make the loudness the same, you would like to modify each track so that it has the same "strength", or norm (technically, unless all the tracks are the same length, you would like them to have the same *norm-squared per time unit*; and to get even more technical, you would also need to make other modifications based upon the frequencies of the input). Thankfully, it is very straightforward to modify a signal so as to change its norm. There are two steps.

First, you will modify the vector so that it is a unit norm, i.e. a norm of 1. This is called **normalizing** the vector. To do that, you simply scale the vector by its norm. Suppose your vector is x; the normalized version of x is then x||x||2. For example, earlier we saw that the vector:

x = [1 2 3]

had a norm of 14. The normalized version of x then is x divided by that value:

$x = [\ 1\ 14\ 2\ 14\ 3\ 14\ ]$

That new vector has a 2-norm of 1, as desired, and is oriented in the exact same direction as the original vector. If you wanted the norm of the vector to be some other value, say the number M, you would multiply the normalized vector by M.

Inner Products and Orthogonality

In discrete-time signal processing, understanding signals as vectors within a vector space allows us to use tools of analysis and linear algebra to examine signal properties. One of the properties we may want to consider is the similarity of (or difference between) two vectors. A mathematical tool that provides insight into this is the **inner product**.

Transposing Vectors

One of the ways to express the inner product of two vectors is through matrix multiplication. To explain that, we must first introduce the concept of transposing vectors. In order to multiply two matrices (a vector is simply a matrix in which one of the dimensions is 1), the column dimension of the first must match the row dimension of the second. To make those match for two vectors with the same dimensions, we must **transpose** one of them. To take the transpose of a matrix, simply turn the rows into columns: the first row will become the first column in the transposed matrix; the second row, the second column, and so on. Here is how that looks for a vector. An N-row, single column vector transposed becomes a 1 row, N-column vector:

$$[x[0]x[1]\vdots x[N-1]]T=[x[0]x[1]\cdots x[N-1]].$$

Now, when it comes to complex-valued vectors, we can take a transpose in the same way, but for the purposes of finding an inner product we actually need to take the conjugate, or Hermitian, transpose, which involves taking the transpose and then the complex conjugate of each entry:

$$\begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix}^H = \begin{bmatrix} x[0]^* & x[1]^* & \cdots & x[N-1]^* \end{bmatrix}$$

Of course, for real valued vectors, the regular transpose and Hermitian transpose are identical.

The Inner Product

The **inner product** of two complex (or real) valued vectors is defined as:

$$\langle x, y \rangle = y^H x = \sum_{n=0}^{N-1} x[n] y[n]^*$$

So the inner product operation takes two vectors as inputs and produces a single number. It turns out that the number it produces is related to the **angle** θ between the two vectors:

$$\cos(\theta_{x,y}) = \frac{\mathrm{Re}\,\langle x, y \rangle}{\|x\|_2 \|y\|_2}$$

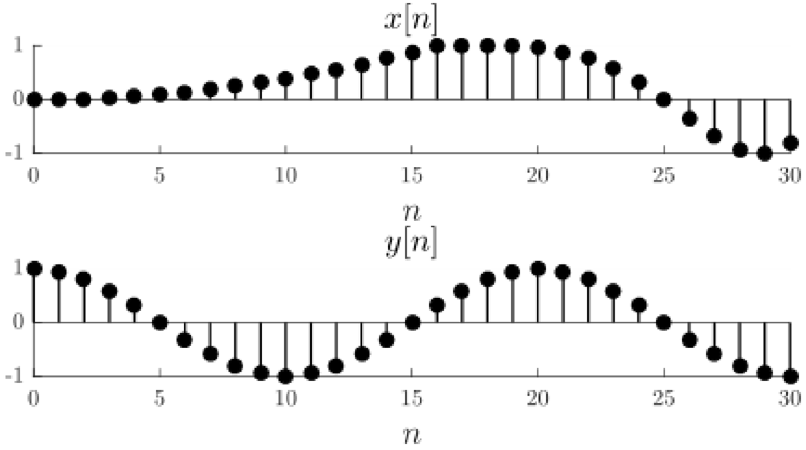
This formula works for complex and real vectors, although taking the real part of the inner product is redundant for real-valued ones.

For two- (or three-) dimensional vectors, this angle is exactly what you would expect it to be. Let

$x = [\ 1\ 2\ ],\quad y = [\ 3\ 2\ ]$

We have $\|x\|22 = 12 + 22 = 5$, $\|y\|22 = 32 + 22 = 13$,and $\langle x,y\rangle = (1)(3) + (2)(2) = 7$. The angle between them is $\arccos(7(5)(13)) \approx 0.519$ rad $\approx 29.7\circ$. If you plot the vectors out in the Cartesian plane, you will indeed see an angle between them of about 30 degrees.

For higher dimensional signals the result of the inner product--how it relates to the angle between signals--may not seem as intuitive, but the information it provides is still just as useful, and of course it is computed in the same was as with shorter vectors. Consider the signals below:



The inner product of these two signals, computed according to the formula above, is $\langle x,y\rangle = yTx = 5.995$, which corresponds to an angle of $\theta_{x,y} = 64.9^\circ$.

The inner product of these two signals is 0. Despite overlapping nonzero entries in the time domain, the inner product of these two particular signals is also 0.

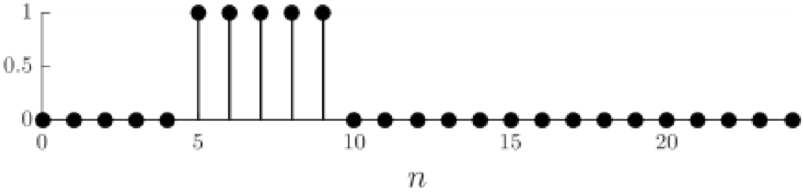
Inner product: Limiting Cases

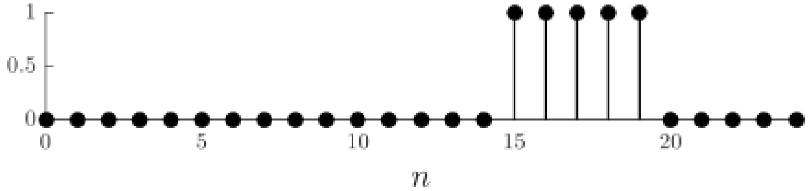
Recall that the inner product is defined as a sum ($\sum_{n=0}^{N-1} x[n]y[n]^*$), which can also be expressed as a vector product ($y^H x$). Let's look at a couple of interesting values that sum could take.

The dot product of two signals could be rather large. If the signals are identical, it is simply the norm of the signal, squared:

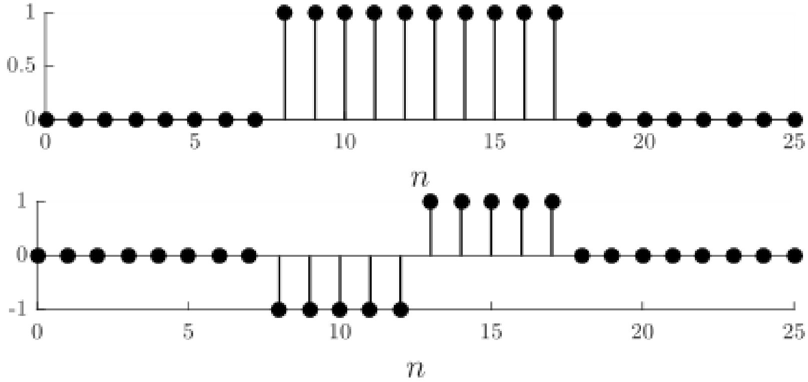
$$\langle x,x\rangle = \sum_{n=0}^{N-1} x[n]x[n]^* = \sum_{n=0}^{N-1} |x[n]|^2 = \|x\|_2^2.$$

On the other hand, it is also possible for the dot product sum to be 0. Consider the two signals below:





The inner product of those two signals is obviously 0 because each point-wise product is also 0. But it is possible, of course, for products in the sum to be nonzero and still have the total add up to 0:



Whenever the inner product of two signals is 0, it is defined that those signals are **orthogonal.**

Orthogonality of Harmonic Sinusoids

Recall the special class of discrete-time finite length signals called harmonic sinusoids:

$s_k[n] = e^{j\frac{2\pi}{N}kn}, \quad n,k,N \in \mathbb{Z}, \ \ 0 \le n \le N-1, \ \ 0 \le k \le N-1$

It is a very interesting property that any two of these sinusoids having different frequencies (i.e., $k \neq l$) are orthogonal:

$$\langle s_k | s_l \rangle = \sum_{n=0}^{N-1} s_k[n] s_l^*[n] = \sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}kn} \left(e^{j\frac{2\pi}{N}ln}\right)^* = \sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}kn} e^{-j\frac{2\pi}{N}ln} = \sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}(k-l)n} \quad \text{let } r = k-l \in \mathbb{Z}, r \neq 0 = \sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}rn} = \sum_{n=0}^{N-1} a^n \quad \text{with} \quad a = e^{j\frac{2\pi}{N}r}, \text{and recall } \sum_{n=0}^{N-1} a^n = \frac{1-a^N}{1-a} = \frac{1-e^{j\frac{2\pi rN}{N}}}{1-e^{j\frac{2\pi r}{N}}} = 0 \quad ✓$$

If two of these sinusoids have the same frequency, then their inner product is simply N:

$$\langle s_k, s_k \rangle = \|s_k\|_2^2 = \sum_{n=0}^{N-1} |s_k[n]|^2 = \sum_{n=0}^{N-1} \left|e^{j\frac{2\pi k}{N}n}\right|^2 = \sum_{n=0}^{N-1} 1 = N ✓$$

So the inner product of two harmonic sinusoids is zero if their frequencies are different, and N if they are the same. In order to make the latter number 1, instead of N, they are sometimes normalized:

$$\tilde{s}_k[n] = \frac{1}{N} e^{j\frac{2\pi}{N}kn}, \quad n, k, N \in \mathbb{Z}, \quad 0 \leq n \leq N-1, \quad 0 \leq k \leq N-1$$

Matrix Multiplication and Inner Products

Let's take look at the formula for the matrix multiplication $y = Xa$. For notation, we will represent the value on the nth row and mth column of X as $x_m[n]$. The matrix multiplication looks like this:

$$y = Xa \quad \begin{bmatrix} \vdots \\ y[n] \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots & \vdots & & \vdots \\ x_0[n] & x_1[n] & \cdots & x_{M-1}[n] \\ \vdots & \vdots & & \vdots \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{M-1} \end{bmatrix}$$

And the value for the multiplication is: $y[n] = \sum_{m=0}^{M-1} \alpha_m x_m[n]$. Hopefully that formula looks familiar! What it is showing is that the matrix multiplication $y = Xa$ is simply the inner product of a with each row of X, when X and a are real. If they are complex-valued, then the complex conjugate of one of them would have to be performed before the matrix multiplication for each value of y to be the inner product of the matrix row with a.

The Cauchy Schwarz Inequality

Above we saw that the inner product of two vectors can be as small as 0, in which case the vectors are orthogonal, or it can be large, such as when the two vectors are identical (and the inner product is simply the norm of the vector, squared). It turns out that there is a very significant inequality that explains these two cases. It is called the **Cauchy Schwarz Inequality**, which states that for two vectors x and y, $|\langle x,y \rangle| \le \|x\| \|y\|$. Now the magnitude of the inner product is always greater than or equal to 0 (being 0 if the vectors are orthogonal), so we can expand the inequality thus: $0 \leq |\langle x,y \rangle| \leq \|x\|\|y\|$. If we divide the equation by $\|x\|\|y\|$, then we have $0 \leq |\langle x,y \rangle| \|x\|\|y\| \leq 1$. This explains why we can define $\cos(\theta x,y) = \mathrm{Re}\ \langle x,y \rangle \|x\|2\|y\|2$, for the cosine function also has a range of 0 to 1.

Now there are many different proofs of the inequality, and it is something of a mathematical pastime to appreciate their various constructions. But for signal processing purposes, we are more interested in the utility of the inequality. What it basically says is that--when the norms of two vectors are taken into consideration--their inner product ranges in value from 0 to 1. Because of this, we can see that the inner product introduces some kind of comparison between two different vectors. It is at its smallest when they are, in a sense, very different from each other, or **orthogonal**. It is at its peak when the vectors are simply scalar multiples of each other, or in other words, are very alike.

It turns out there are many applications in which we would like to determine how similar one signal is to another. How does a digital communication system decide whether the signal corresponding to a "0" was transmitted or the signal corresponding to a "1"? How does a radar or sonar system find targets in the signal it receives after transmitting a pulse? How does a computer vision system find faces in images? For each of these questions, the similarity/

dissimilarity bounds established by the Cauchy Schwarz inequality help us to determine the answer.

Norms and Inner Products of Infinite Length Vectors

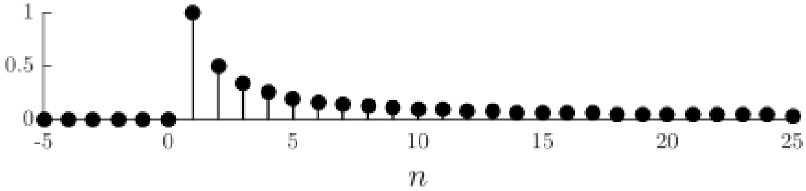
As with finite length signals (which can be understood as vectors in RN or CN), we can take the norms and inner products of infinite length signals, as well. For the most part, the concepts will apply just the same.

The Norms of Infinite Length Vectors

As with finite-length vectors, the p-norm for infinite-length vectors is a sum: $\|x\|_p = \left(\sum_{n=-\infty}^{\infty} |x[n]|^p\right)^{1/p}$, $p \geq 1$. Unlike the p-norms of finite-length signals, the summation for infinite-length vectors is not necessarily finite. As a consequence, the p-norms of infinite length signals may not be bounded. Consider a finite-length signal $x[n] = 1$ for $0 \leq n < N-1$. The 2-norm for this signal would be N. However, for an infinite length signal of constant value 1, all the p-norms will be unbounded!

For some infinite-length vectors, the p-norm may exist only for certain values of p. Consider, for example, the signal:

$$x[n] = \begin{cases} 0 & n \leq 0 \\ \frac{1}{n} & n \geq 1 \end{cases}$$



The 2-norm of this signal exists:

$\|x\|_2^2 = \sum_{n=-\infty}^{\infty} |x[n]|^2 = \sum_{n=1}^{\infty} \left|\frac{1}{n}\right|^2 = \sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6} \approx 1.64.$

But the 1-norm is unbounded: $\|x\|_1 = \sum_{n=-\infty}^{\infty} |x[n]| = \sum_{n=1}^{\infty} \frac{1}{n} = \infty$.

For a finite-length vector, the ∞-norm is simply the maximum value of the magnitudes of all its elements. For infinite-length vectors, it is instead the **supremum**, or the **least upper bound** of the set of all the elements of the vector. The distinction between that value and the maximum value of the set is outside the bounds of this course. For all the cases we will consider, the ∞-norm will simply be the largest of the magnitudes of the elements of the vector. For example, for the signal $x[n]$ above, $\|x[n]\|_\infty = 1$.

Inner Product of Infinite Length Signals

The inner product of infinite length signals is defined the same way as for finite-length signals, except that the sum is infinite:

$\langle x,y\rangle = \sum_{n=-\infty}^{\infty} x[n]y[n]^*$. The relationship to the angle between the signals is also the same: $\cos\theta_{x,y} = \frac{\mathrm{Re}\,\langle x,y\rangle}{\|x\|_2\|y\|_2}$.

Linear Combinations of Infinite Length Vectors

When it comes to infinite-length vectors, we will also be interested in linear combinations of them. However, in contrast with finite-length vectors, we will be interested in the linear combination of an infinite number of the vectors: $y = \sum_{m=-\infty}^{\infty} \alpha_m x_m$. As with finite-length vectors, infinite-length vectors can be stacked into a matrix (an infinitely large one) and be multiplied by an infinite length vector of scalers to perform linear combination:

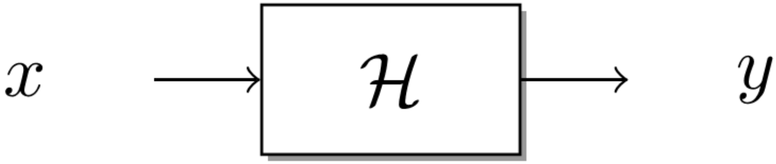
$$X = \left[\,\cdots \mid x_{-1} \mid x_0 \mid x_1 \mid \cdots\,\right]$$

$$a = \begin{bmatrix} \vdots \\ \alpha_{-1} \\ \alpha_0 \\ \alpha_1 \\ \vdots \end{bmatrix}$$

$$y = \sum_{m=-\infty}^{\infty} \alpha_m x_m = \begin{bmatrix} & \vdots & \vdots & \vdots & \\ \cdots & x_{-1}[-1] & x_0[-1] & x_1[-1] & \cdots \\ \cdots & x_{-1}[0] & x_0[0] & x_1[0] & \cdots \\ \cdots & x_{-1}[1] & x_0[1] & x_1[1] & \cdots \\ & \vdots & \vdots & \vdots & \end{bmatrix} \begin{bmatrix} \vdots \\ \alpha_{-1} \\ \alpha_0 \\ \alpha_1 \\ \vdots \end{bmatrix} = Xa$$

Derived copy of Discrete-time Systems

Discrete-time signals are mathematical entities; in particular, they are functions with an independent time variable and a dependent variable that typically represents some kind of real-world quantity of interest. But as interesting as a signal may be on its own, engineers usually want to *do* something to it. This kind of action is what discrete-time systems are all about. A **discrete-time system** is a mathematical transformation that maps a discrete-time input signal (usually designated x) to a discrete-time output signal (usually designated y). In other words, it takes an input signal and modifies it to produce an output signal:
System H takes takes a discrete time signal x as an input and produces an output y.



There is no end to the possibilities of what a system could do. Systems might be trivially dull (e.g., produce an output of 0 regardless of the input) or incredibly complex (e.g., isolate a single voice speaking in a crowd). Here are a few examples of systems:

- A speech recognition system converts acoustic waves of speech into text

- A radar system transforms the received radar pulse to estimate the position and velocity of targets
- A functional magnetic resonance imaging (fMRI) system transforms measurements of electron spin into voxel-by-voxel estimates of brain activity
- A 30 day moving average smooths out the day-to-day variability in a stock price

Signal Length and Systems

Recall that discrete-time signals can be broadly divided into two classes based upon their length: they are either infinite-length or finite-length (and recall also that periodic signals, though infinite in length, can be viewed as finite-length signals when we take a single period into account). Likewise, discrete-time systems are also finite- or infinite-length, depending on the kind of input signals they take. Finite-length systems take in a finite-length input and produce a finite-length output (of the same length), with infinite-length systems doing the same for infinite-length signals.

Examples of Discrete-time Systems

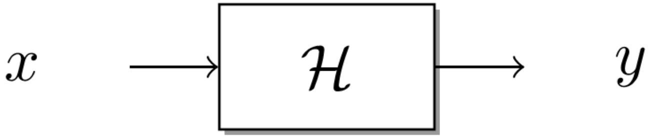
So a system takes an input signal x and produces an output signal y. How does this look, mathematically? Below are several examples of systems and their mathematical expression:

- Identity: $y[n]=x[n]$
- Scaling: $y[n]=2x[n]$
- Offset: $y[n]=x[n]+2$
- Square signal: $y[n]=(x[n])2$
- Shift: $y[n]=x[n+m], m \in Z$
- Decimate: $y[n]=x[2n]$
- Square-time: $y[n]=x[n2]$
- Moving average (combines shift, sum, and scale): $y[n]=12(x[n]+x[n-1])$
- Recursive average: $y[n]=x[n]+\alpha y[n-1]$

So systems take input signals and produce output signals. We have seen some examples of systems, and have also introduced a broad categorization of systems as either operating on finite- or infinite length signals.

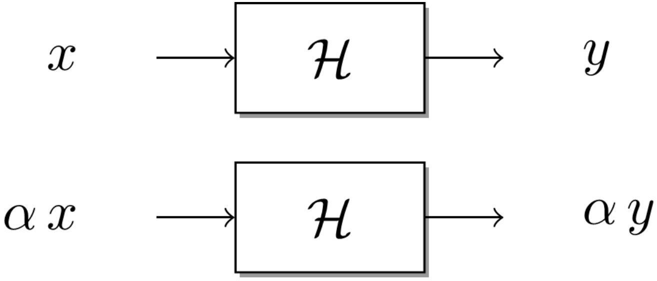
Linear Systems

Discrete-time systems are mathematical transformations that take input signals and map them to output signals. For a given input x, a discrete-time system will produce a new signal y:

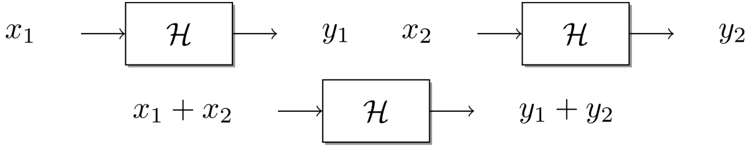


It turns out that it is very important to know or determine if a system has certain characteristics, and among these is the classification of linearity. A system is **linear** if it has two special properties: *scaling* (sometimes called homogeneity) and *additivity*. The first of these is satisfied if, for any arbitrary input x, scaling the input by any complex valued constant value α will result in the output being scaled by the same value. Mathematically, this scaling rule is represented as $H\{\alpha x\} = \alpha H\{x\}$, and as a system diagram it looks like this:

A system that satisfies the scaling property.



A system has the additivity property if, for any two arbitrary inputs, the output of the sum of them is the same as the sum of their individual outputs: $H\{x1 + x2\} = H\{x1\} + H\{x2\}$.
A system that satisfies the additivity property.



A system has both the additivity and scaling properties (and thus, by definition, is linear) if for arbitrary inputs x1 and x2 and arbitrary constants $\alpha1$ and $\alpha2$, $H\{\alpha1x1 + \alpha2x2\} = \alpha1H\{x1\} + \alpha2H\{x2\}$. If a system lacks either of the scaling or additivity property, then it is said to be **nonlinear**.

Determining Linearity

Consider again the definition of linearity; to be linear, a system must preserve the scaling and additivity properties for any arbitrary input. Therefore, determining linearity amounts to completing a mathematical proof that assumes arbitrary inputs and scaling factors, and then shows the necessary result. Suppose $H\{x[n]\} = 3x[n]$. Here is how the additivity proof would look:

Let x1 and x2 be arbitrary inputs to system H and let $\alpha1$ and $\alpha2$ be arbitrary constants.

$H\{x1[n]\}=3x1[n]H\{x2[n]\}$
$=3x2[n]H\{\alpha1x1[n]+\alpha2x2[n]\}$
$=3(\alpha1x1[n]+\alpha2x2[n])=3\alpha1x1[n]+3\alpha2x2[n]=\alpha1(3x1[$
$\rightarrow$Linear.

Now, to show a system is nonlinear requires a different kind of proof. Rather than having to prove both of the properties hold for any arbitrary input(s), only a single example needs to be provided for which either of the properties fail. For example, consider the system $H\{x[n]\}=x[n]+1$. We can show it is nonlinear by considering just the single following case:

Let $x[n]=0$ $H\{x[n]\}=x[n]+1=0+1=1$ But $H\{2x[n]\}=2x[n]+1=2\cdot0+1=1\neq2H\{x[n]\}\rightarrow$ Nonlinear

> Good students of signals and systems must become adept at determining the linearity (or nonlinearity) of systems. Practice on the system examples below; which of them are linear?
>
> - Identity: $y[n]=x[n]$
> - Scaling: $y[n]=2x[n]$
> - Offset: $y[n]=x[n]+2$
> - Square signal: $y[n]=(x[n])2$
> - Shift: $y[n]=x[n+m]m\in Z$
> - Decimate: $y[n]=x[2n]$
> - Square time: $y[n]=x[n2]$
> - Moving average (combines shift, sum,

scale): $y[n]=12(x[n]+x[n-1])$

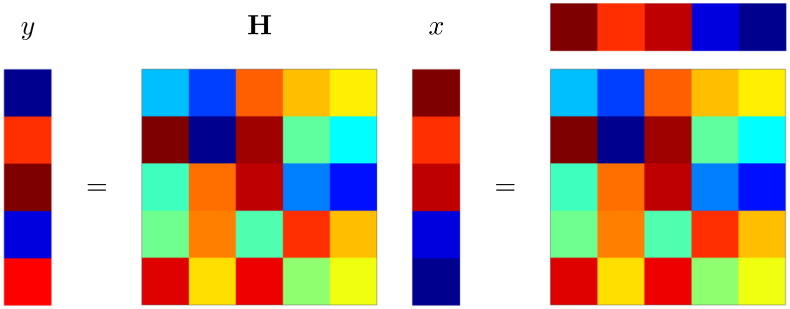
- Recursive average: $y[n]=x[n]+\alpha y[n-1]$

-
  - Identity: **Linear**
  - Scaling: **Linear**
  - Offset: **Nonlinear**
  - Square signal: **Nonlinear**
  - Shift: **Linear**
  - Decimate: **Linear**
  - Square time: **Linear**
  - Moving average: **Linear**
  - Recursive average: **Linear**

Linear Systems and Matrix Multiplication

One of the interesting characteristics of any linear system is that its input/output relationship can be expressed as a matrix multiplication (note that this is distinct from--though ultimately related to--the notion of using matrix multiplication to express one signal as a linear combination of others). In fact, this relationship is actually an identity: any linear system can be expressed as a matrix multiplication, and matrix multiplications are linear systems. Below is how to represent any linear system mathematically, either in matrix multiplication notation, $y = Hx$ or, in summation notation, y[n] = Σm[H]n,mx[m] = Σmhn,mx[m] (where hn,m = [H]n,m represents the row-n, column-m entry of the matrix H).

This matrix multiplication can be understood in two ways. First, the multiplication means that each value in the vector y is the inner product of the corresponding row of H with the vector x. Or, equivalently, the vector y can be seen as a weighted sum of the columns of H, with the values in the vector x being the weights of the corresponding columns. Below is a picture of matrix multiplication, with different colors representing different values. Try to comprehend the multiplication with both perspectives.

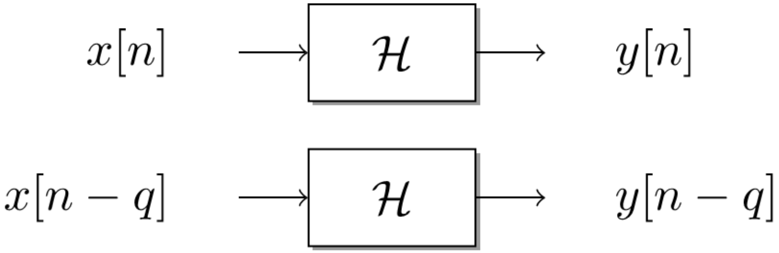


Time-Invariant Systems

Recall that a discrete-time system is a mathematical entity that takes an input signal (usually denoted x) and produces an output signal (usually denoted y). In the study of signal processing, systems that have certain characteristics are of particular interest. Among these characteristics is **time-invariance**.

There is a very basic intuition behind the notion of time-invariance. In many cases, we would like a system to behave a certain way, no matter when an input may be given. To use a practical analogy, people expect their toasters to operate the same way on Tuesdays as they do on Mondays.

A system is **time-invariant** if a time delay for an input results in the same time delay on the output. Expressing this idea of system time-invariance mathematically is straightforward. Consider a system H; let x[n] be some arbitrary input, and call the system's output for that input y[n]. H is time-invariant if, for some arbitrary integer value q, $H[x[n-q]] = y[n-q]$:



Example: Moving Average System

Consider, for example, the moving average system $y[n] = 12(x[n] + x[n-1])$. Is this system time invariant? To find out, let's delay the input by some value q, and see what the output is. We'll call the delayed input signal $x'[n]$ $(= x[n-q])$ and the new output y':

$y'[n] = 12(x'[n] + x'[n-1]) = 12(x[n-q] + x[(n-1)-q])$.

Now the important question, is $y'[n] = y[n-q]$? That will be easy to determine, simply replace every n with $n-q$ in that original $y[n]$ equation and see if it is the same as $y'[n]$:$y[n-q] = 12(x[n-q] + x[n-q-1]) = 12(x[n-q] + x[(n-1)-q]) = y'[n]$✓

Example: Decimation

Now let's consider a decimation system, $y[n] = x[2n]$. Is this system time-invariant? First, create a new signal that is a delayed version of $x[n]$: $x'[n] = x[n-q]$. Next, find the output corresponding to this new signal: $y'[n] = x'[2n]$. Now express this output in terms of the original input: $y'[n] = x'[2n] = x[2n-q]$. Finally, check to see if this is the same as $y[n-q]$. As $y[n-q]=x[2(n-q)]$ is not equivalent to $y'[n]$ (since $2n-1 \neq 2(n-q)$ for all n and q), the system is *not* time-invariant.

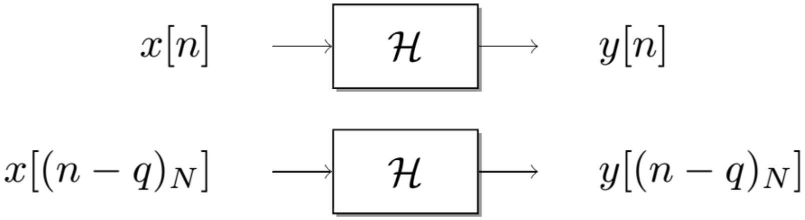
> Now practice on the system examples below; which of them are time-invariant?

- Identity: $y[n]=x[n]$
- Scaling: $y[n]=2x[n]$
- Offset: $y[n]=x[n]+2$
- Square signal: $y[n]=(x[n])2$
- Shift: $y[n]=x[n+m]m\in Z$
- Square time: $y[n]=x[n2]$
- Recursive average: $y[n]=x[n]+\alpha y[n-1]$

- Identity: **time-invariant**
- Scaling: **time-invariant**
- Offset: **time-invariant**
- Square signal: **time-invariant**
- Shift: **time-invariant**
- Square time: **not time-invariant**
- Recursive average: $y[n]=x[n]+\alpha y[n-1]$**time-invariant**

Time-Invariance for Finite-Length Systems

All of the discussion above has considered systems which take infinite-length signals as inputs. A system which takes finite-length signals as inputs and produces them as outputs can also exhibit the characteristic of time-invariance, albeit with a slightly different definition. Finite-length systems are time-invariant if any *circular* shift on the input results in a corresponding *circular* shift on the output:



Linear Time-Invariant Systems

Among the many characteristics and classifications of discrete-time systems, two of particular importance are linearity and time-invariance. If a system happens to exhibit *both* of these qualities, then it is referred to as being an **LTI system** (linear time-invariant). These systems are very significant in the study of signal processing, for reasons that will be clear when the concept of the system impulse response is considered. Using the tests for both linearity and time-invariance, determine whether or not the following systems are LTI.

Consider the systems below. Which are LTI?

- Identity: $y[n]=x[n]$
- Scaling: $y[n]=2x[n]$
- Offset: $y[n]=x[n]+2$
- Square signal: $y[n]=(x[n])2$
- Shift: $y[n]=x[n+m]m\in Z$
- Decimate: $y[n]=x[2n]$
- Square time: $y[n]=x[n2]$
- Moving average (combines shift, sum, scale): $y[n]=12(x[n]+x[n-1])$
- Recursive average: $y[n]=x[n]+\alpha y[n-1]$

- Identity: **LTI**
- Scaling: **LTI**

- Offset: **not LTI (time-invariant, but not linear)**
- Square signal: **not LTI (time-invariant, but not linear)**
- Shift: **LTI**
- Decimate: **not LTI (linear, but not time-invariant)**
- Square time: **not LTI (time-invariant, but not linear)**
- Moving average (combines shift, sum, scale): **LTI**
- Recursive average: **LTI**

A pictorial representation of a Toeplitz matrix.

Matrix Structure of LTI Systems (infinite-length)

We recall that linear systems can be expressed through a matrix multiplication. Suppose that a system is linear; any output y can be expressed as the multiplication of an infinite-dimensional matrix H with the input x:

$$y = H x \quad y[n] = \Sigma m [H] n, m x[m]$$

Now if a linear system is also time-invariant, it turns out the matrix H will have an interesting structure. To see this, we will first express the matrix multiplication in a summation notation, where

$h_{n,m} = [H]_{n,m}$ represents the row-n, column-m entry of the matrix H:

$$y[n] = H\{x[n]\} = \sum_{m=-\infty}^{\infty} h_{n,m} x[m], \quad -\infty < n < \infty$$

Supposing the system is time-invariant, we have: $H\{x[n-q]\} = \sum_{m=-\infty}^{\infty} h_{n,m} x[m-q] = y[n-q]$

If we apply a simple change of variables ($n'=n-q$ and $m'=m-q$), we then have:

$H\{x[n']\} = \sum_{m'=-\infty}^{\infty} h_{n'+q,m'+q} x[m'] = y[n']$.

If we compare this final equation with the original one, we see that for an LTI system, $h_{n,m} = h_{n+q,m+q} \forall q \in Z$.

So for LTI systems, the matrix H that defines the system's input/output relationship has a special structure:

$$h_{n,m} = h_{n+q,m+q} \; \forall q \in Z$$

(where $h_{n,m}$ is the value at the nth row and mth column of the matrix H). Such matrices are called **Toeplitz matrices**. They have identical values along their diagonals:

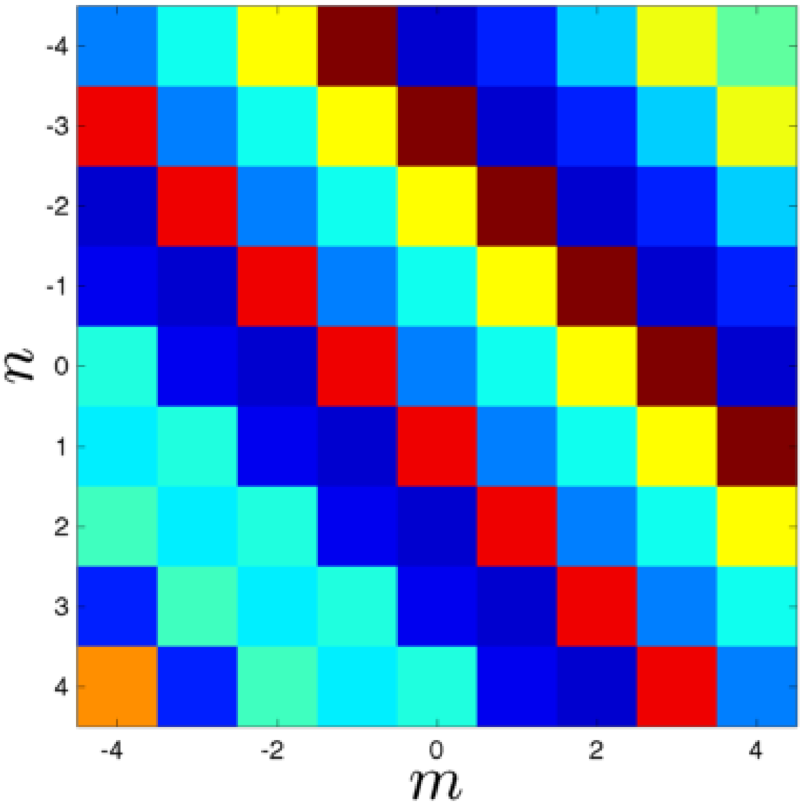
H = [$\vdots \; \vdots \; \vdots$ $\cdots$ $h_{-1,-1}$ $h_{-1,0}$ $h_{-1,1}$ $\cdots$ $\cdots$ $h_{0,-1}$ $h_{0,0}$ $h_{0,1}$ $\cdots$ $\cdots$ $h_{1,-1}$ $h_{1,0}$ h_{1}

, 1 ⋯ ⋮ ⋮ ⋮] = [⋮ ⋮ ⋮ ⋯ h 0 , 0 h − 1 , 0 h − 2 , 0 ⋯ ⋯ h 1 , 0 h 0 , 0 h − 1 , 0 ⋯ ⋯ h 2 , 0 h 1 , 0 h 0 , 0 ⋯ ⋮ ⋮ ⋮]

Because of that property, all of the information in the matrix is contained in a single row, or column. We'll call the 0th row h[n]: h[n] = hn,0:

H = [⋮ ⋮ ⋮ ⋯ h 0 , 0 h − 1 , 0 h − 2 , 0 ⋯ ⋯ h 1 , 0 h 0 , 0 h − 1 , 0 ⋯ ⋯ h 2 , 0 h 1 , 0 h 0 , 0 ⋯ ⋮ ⋮ ⋮] = [⋮ ⋮ ⋮ ⋯ h [0] h [− 1] h [− 2] ⋯ ⋯ h [1] h [0] h [− 1] ⋯ ⋯ h [2] h [1] h [0] ⋯ ⋮ ⋮ ⋮]

Below is an example of a pictorial representation of a Toeplitz matrix. Note how the diagonals are all the same color:



A pictorial representation of a circulant matrix.

Matrix Structure of LTI Systems (finite-length)

For systems operating on finite-length signals, we can apply similar analysis. Linear systems can be expressed as a matrix multiplication $y=Hx$, here shown using summation notation: $y[n] = H\{x[n]\} = \Sigma m=0N-1hn,mx[m], 0 \leq n \leq N-1$. If we enforce time-invariance on such a system, we then have:

$$H\{x[(n-q)_N]\} = \sum_{m=0}^{N-1} h_{n,m} x[(m-q)_N] = y[(n-q)_N].$$

A change of variables ($n' = n-q$) then yields:

$$H\{x[(n')_N]\} = \sum_{m'=-q}^{M-1-q} h_{(n'+q)_N,(m'+q)_N} x[(m')_N] = y[(n')_N]$$

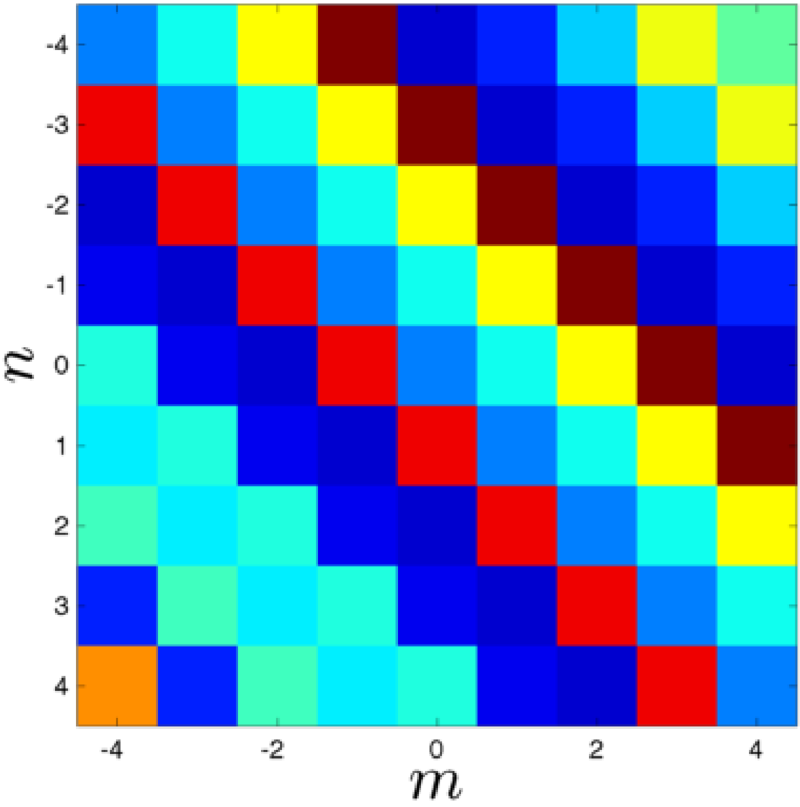
We see, then, that for LTI finite-length systems, $h_{n,m} = h_{(n+q)_N,(m+q)_N} \forall q \in \mathbb{Z}$. This is similar to the Toeplitz structure for infinite-length systems, but note the circular shifts, how the rows "wrap" around the edges of the matrix:

$$H = \begin{bmatrix} h_{0,0} & h_{0,1} & h_{0,2} & \cdots & h_{0,N-1} \\ h_{1,0} & h_{1,1} & h_{1,2} & \cdots & h_{1,N-1} \\ h_{2,0} & h_{2,1} & h_{2,2} & \cdots & h_{2,N-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ h_{N-1,0} & h_{N-1,1} & h_{N-1,2} & \cdots & h_{N-1,N-1} \end{bmatrix} = \begin{bmatrix} h_{0,0} & h_{N-1,0} & h_{N-2,0} & \cdots & h_{1,0} \\ h_{1,0} & h_{0,0} & h_{N-1,0} & \cdots & h_{2,0} \\ h_{2,0} & h_{1,0} & h_{0,0} & \cdots & h_{3,0} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ h_{N-1,0} & h_{N-2,0} & h_{N-3,0} & \cdots & h_{0,0} \end{bmatrix}$$

This kind of matrix is called a **circulant matrix**. As with the Toeplitz matrices, all of the information in a circulant matrix is contained in a single row or column, the other rows/columns simply being (circularly) shifted versions of the original. So we can call the N-length signal h[n] the 0th column of the matrix, and express the whole matrix in terms of it:

$$H = \begin{bmatrix} h_{0,0} & h_{N-1,0} & h_{N-2,0} & \cdots & h_{1,0} \\ h_{1,0} & h_{0,0} & h_{N-1,0} & \cdots & h_{2,0} \\ h_{2,0} & h_{1,0} & h_{0,0} & \cdots & h_{3,0} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ h_{N-1,0} & h_{N-2,0} & h_{N-3,0} & \cdots & h_{0,0} \end{bmatrix} = \begin{bmatrix} h[0] & h[N-1] & h[N-2] & \cdots & h[1] \\ h[1] & h[0] & h[N-1] & \cdots & h[2] \\ h[2] & h[1] & h[0] & \cdots & h[3] \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ h[N-1] & h[N-2] & h[N-3] & \cdots & h[0] \end{bmatrix}$$

Below is a pictorial representation of a circulant matrix. Like Toeplitz matrices, the values along diagonals are equivalent, but in addition to having this property, the values also "wrap around" the matrix boundaries:

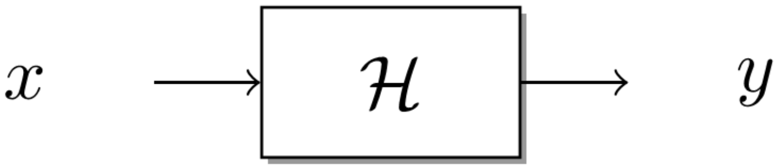


The Impulse Response of Discrete-Time Systems

The system H takes an input x and produces an output y. The system matrix of an LTI system has a Toeplitz structure. Note how when the H matrix is multiplied by it, the delta function "selects" the 0th column. A pictorial representation of the impulse response and system matrix of a finite-length LTI system.A recursive average system ($y[n]=x[n]+\alpha y[n-1]$) is an example of an IIR system, as the impulse response has an infinite number of nonzero values. The impulse response of the moving average system above ($y[n]=12(x[n]+x[n-1])$) has only 2 nonzero values, and thus is an example of an FIR system.

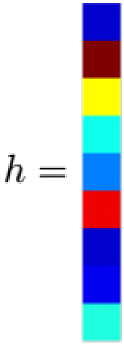
The Impulse Response of Infinite-Length LTI Systems

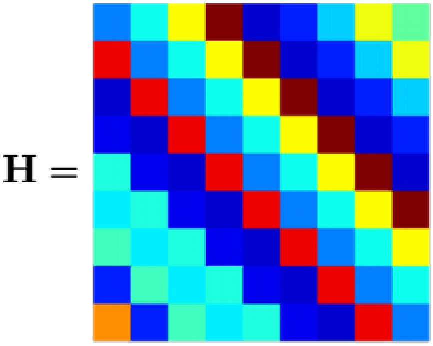
Systems are mathematical transformations that take input signals and map them to output signals:



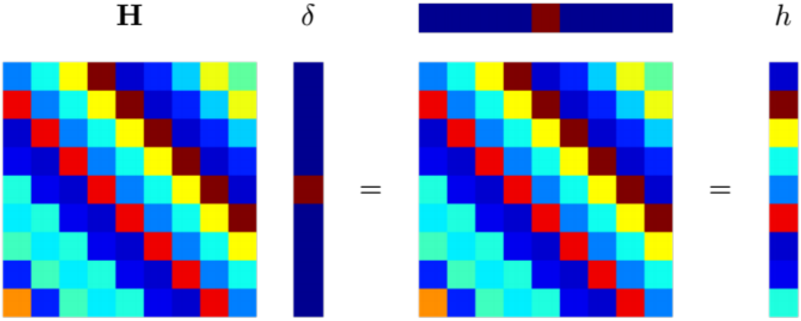
Recall that linear systems can be expressed as matrix multiplications, the output y being the product of the infinitely long matrix H and the input vector x: $y=Hx$. Recall also that if a linear system is also time-invariant, the matrix H has a special

"Toeplitz" structure: each column/row is simply a shifted version of any other column/row.





The 0th column of the H matrix for an LTI system has a special interpretation. If we multiply the H matrix by a delta function, then the result will simply be that 0th column:



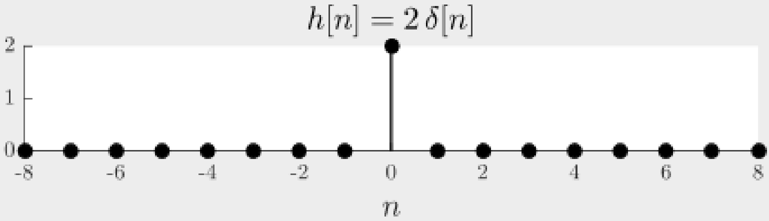
We call the result of this multiplication, i.e. the multiplication of the H matrix by delta (impulse) function, the **impulse response** of the system. It is usually denoted h[n], or if referring to it as a vector, as h. The impulse response h[n] is the output of a system when the input is δ[n]. Using the matrix notation from above, h is the 0th column of the system matrix H.

Because the impulse response is simply a column (specifically, the 0th column) of the system matrix H, then if H is an LTI system, the impulse response completely characterizes the system. It tells us absolutely everything we need to know about the system! Why is this? Because we can construct the entire system matrix H by simply shifting the impulse response accordingly. The value of the matrix at row n and column m is $h[n-m]$.

As a result, we can also express the system input-output matrix multiplication in terms of the impulse response. If we have that $[H]_{n,m}$ is the value of the H matrix at row n and column m, then the matrix multiplication $y[n] = Hx = \sum_{m=-\infty}^{\infty} [H]_{n,m} x[m]$. From above we have the matrix values expressed in terms of the impulse response; subbing this in, we see the output is a sum involving the impulse response and the input: $y[n] = Hx = \sum_{m=-\infty}^{\infty} h[n-m] x[m]$.

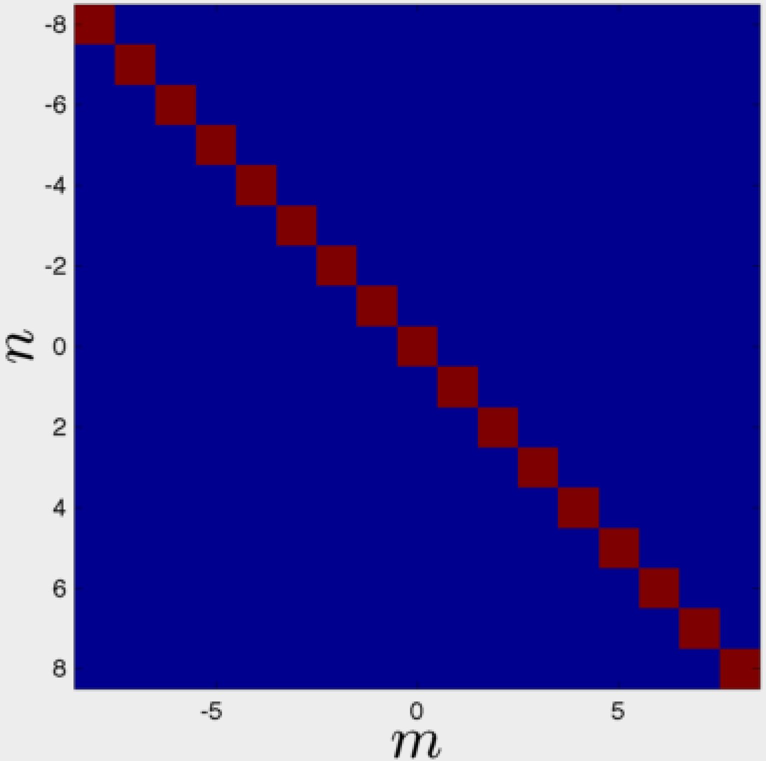
Impulse Response of the Scaling System

Let's find the impulse response of a simple scaling system: $y[n] = H\{x[n]\} = 2x[n]$. To find the impulse response $h[n]$, simply calculate the output when the input is a delta function: $h[n] = H\{\delta[n]\}) = 2\delta[n]$.



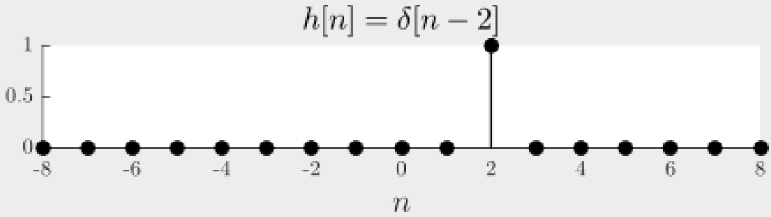
If we would like to know what the system matrix H is, we use the formula [H]n,m = h[n − m] = 2δ[n − m]. Below is a pictorial representation of that matrix:

The system matrix of a simple scaling system.



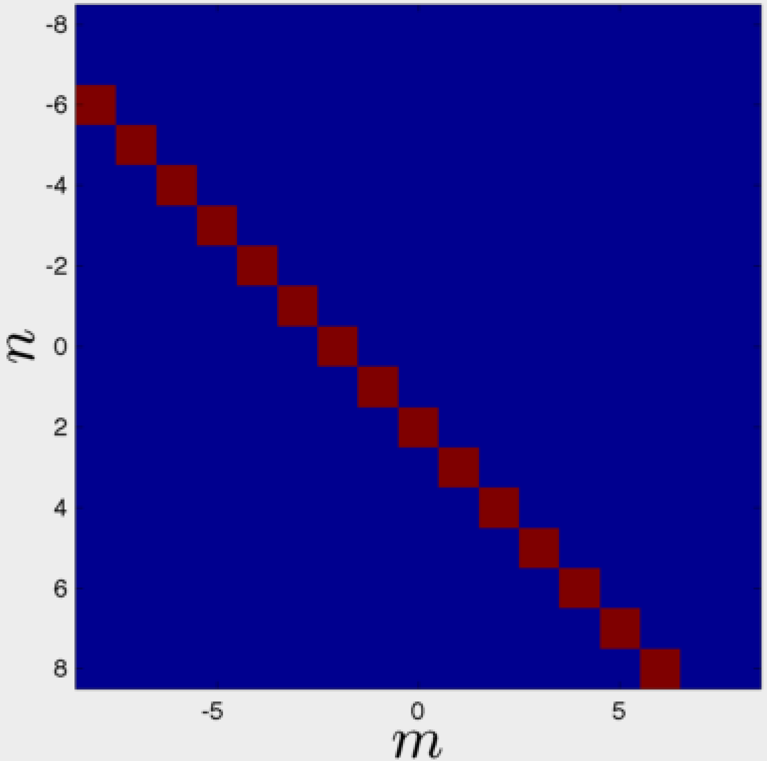
Impulse Response of the Shift System

Now let's try finding the impulse response of a simple time-shifting system: $y[n]=H\{x[n]\}=x[n-2]$. Once again, find the impulse response by supplying a delta function as the input: $h[n]=H\{\delta[n]\}=\delta[n-2]$.



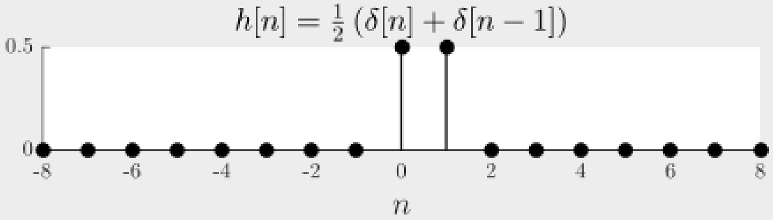
The system matrix H can be found, as before, as $[H]n,m=h[n-m]=\delta[n-m-2]$.

The system matrix of a shift system.



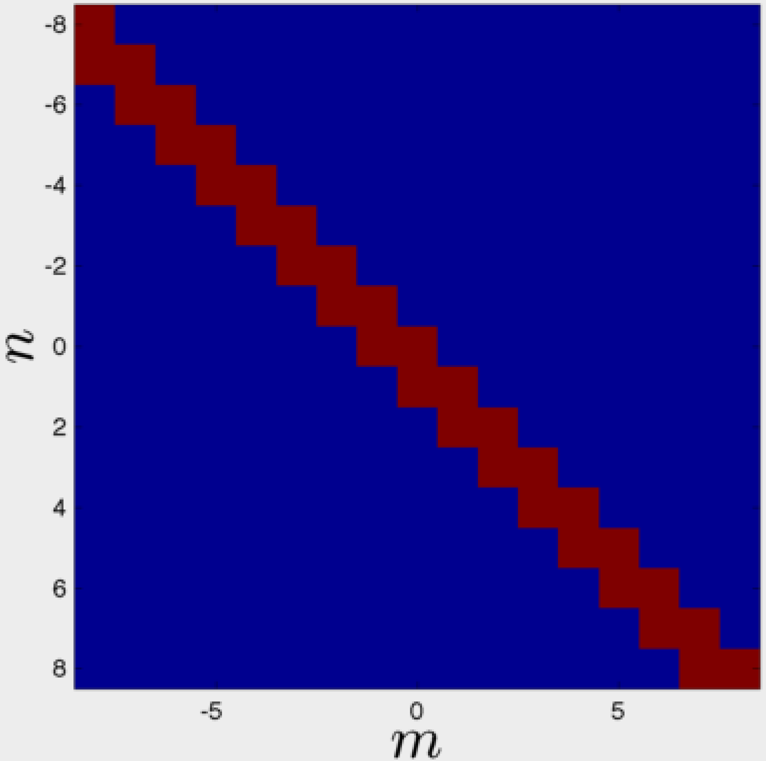
Impulse Response of a Moving Average System

Consider now a moving average system: $y[n] = H\{x[n]\} = 12(x[n]+x[n-1])$. For this system, we find that the impulse response is: $h[n] = H\{\delta[n]\} = 12(\delta[n]+\delta[n-1])$.



The system matrix is found to be $[H]n,m=h[n-m]=12(\delta[n-m]+\delta[n-m-1])$.

The system matrix of a moving average system.

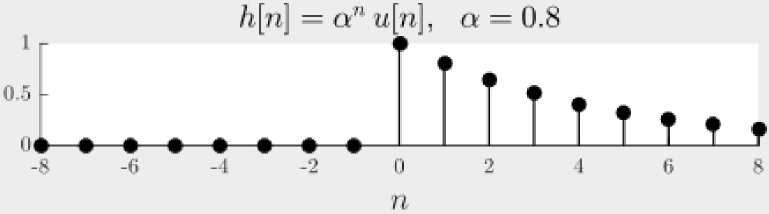


Impulse Response of the Recursive Average System

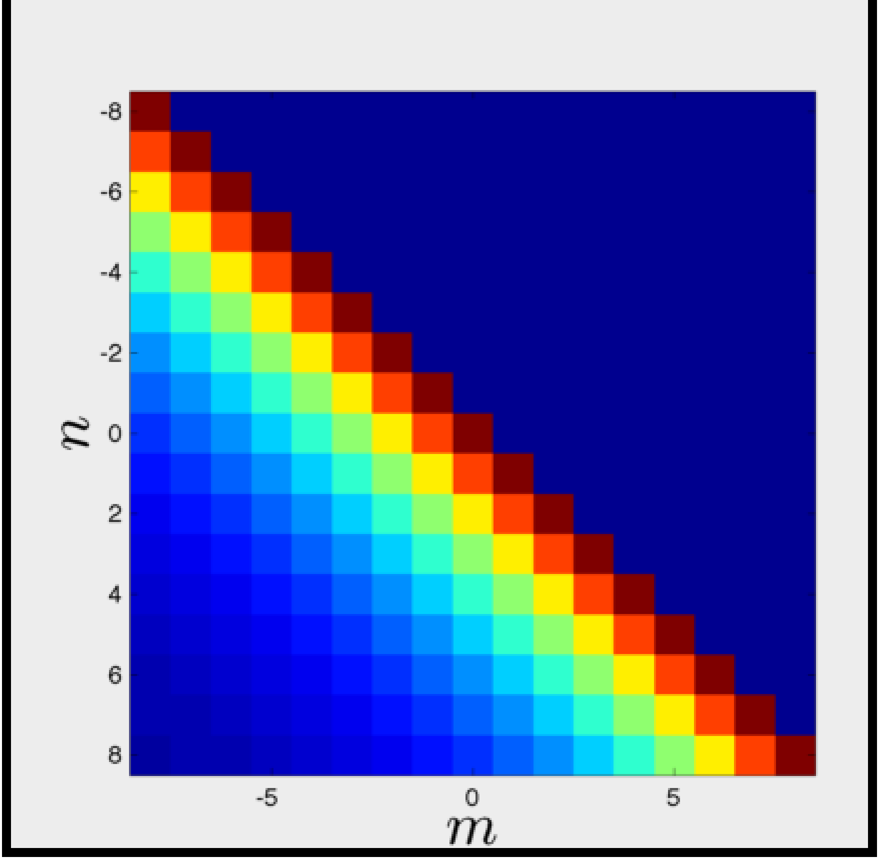
A recursive moving average system has the following input-output relationship:$y[n] = H\{x[n]\} = x[n] + \alpha y[n-1]$. To find the impulse response of this system, we first let $x[n] = \delta[n]$ and find the output: $h[n] = H\{x[n]\} + \alpha h[n-1]$. A bit more work is required if we would like a non-recursively defined impulse response. Assuming that the system is initially at rest (it has no output in the absence of an input), then we can find the output by computing several values of h[n]:

$h[-1] = 0 h[0] = \delta[0] + \alpha h[-1] = 1 + \alpha \cdot 0 = 1 h[1] = \delta[1] +$

Having discerned a pattern, we see that $h[n] = \alpha n u[n]$ (this may be more rigorously proven by induction, if desired):



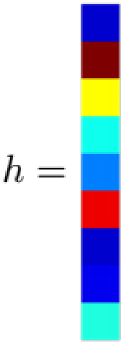
A pictorial representation of the recursive average system.

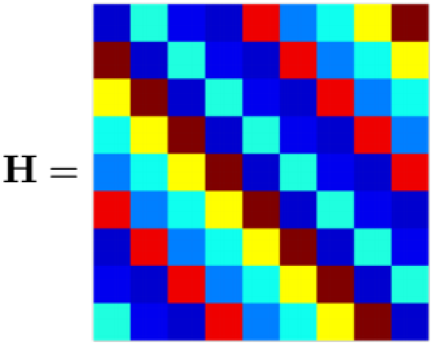


The Impulse Response of Finite-Length LTI Systems

For a finite-length LTI system, just as with an infinite-length system, each row/column of its system matrix is a shifted version of another row/column. Thus, the system matrix for a finite-length system is also of a Toeplitz structure. However, for a finite-length system, not only are the rows/columns shifted, they are *circularly* shifted; the values in the rows/columns "wrap around" when shifted across the boundary of the matrix.

As a result, the entire system matrix can be expressed in terms of the 0th column of the matrix, which (as with infinite-length systems) is given the mathematical expression h[n] and is named the impulse response. For a length-N LTI system, the formula to determine the value of the matrix at a particular row n and column m is $[H]_{n,m} = h[(n - m)_N]$. Note how this is nearly identical to the infinite length formula, except for the modulo-N operator.



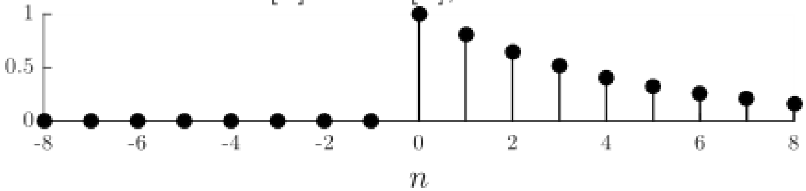


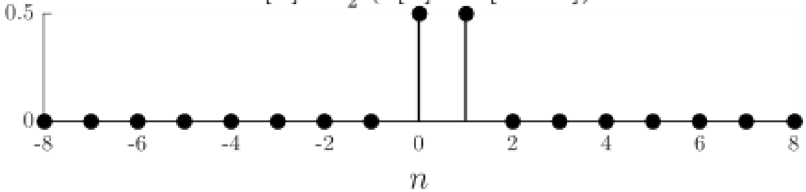
As with infinite-length systems, being able to express the entire system matrix in terms of the impulse response means that the system's input-output matrix multiplication relationship can be expressed as a sum involving the input and impulse response:

$$y[n] = Hx = \Sigma_{m=0}^{N-1} h[(n-m)_N]\, x[m].$$

Impulse Response Length of LTI Systems

For LTI systems, it is often of interest whether its impulse response is of infinite length (i.e., infinite nonzero support) or finite length. Such a distinction is typically made only for infinite-length systems (as the impulse response of finite-length systems is obviously always finite-length!). If a system has an impulse response of infinite length, it is called an **infinite impulse response (IIR)** system; if its impulse response is of finite length, it is called a **finite impulse response (FIR)** system.


$h[n] = \alpha^n\, u[n], \quad \alpha = 0.8$


$h[n] = \frac{1}{2}\left(\delta[n] + \delta[n-1]\right)$

Convolution of Infinite-Length Discrete-Time Signals

It is a fundamental operation in signal processing, given an LTI system H and input x[n], to find the output y[n]. From our study of LTI systems, we have seen that there are several ways to find the output of the system:

- One option is to find the output y[n] at each time n using the input-output relationship (which defines the output y[n] in terms of the input and output at other various times).
- Another option is to take the system matrix H and the input vector x and perform a matrix multiplication $y=Hx$.
- A final option is to find the output at a given time in terms of a summation involving only the input and the system's impulse response: $y[n]\ =\ \Sigma m=-\infty\infty h[n-m]x[m]$.

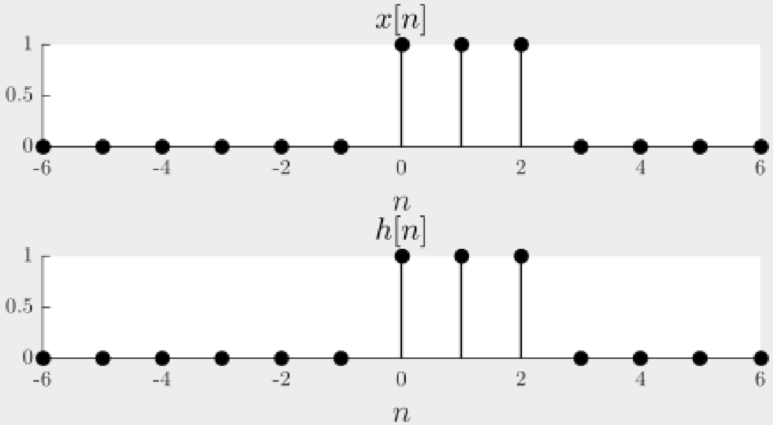
That final option has a special name: it is the **convolution** of x[n] and the system impulse response h[n]. It also has a special operator symbol, $*$, so that the convolution of x[n] and h[n] is expressed as $x[n]*h[n]$.

Computing Convolution

Given an LTI system's impulse response h[n] and an input x[n], the process of computing the output y[n] via the convolution sum formula ($\Sigma_{m=-\infty}^{\infty} h[n-m]x[m]$) is straightforward, according to the following steps:

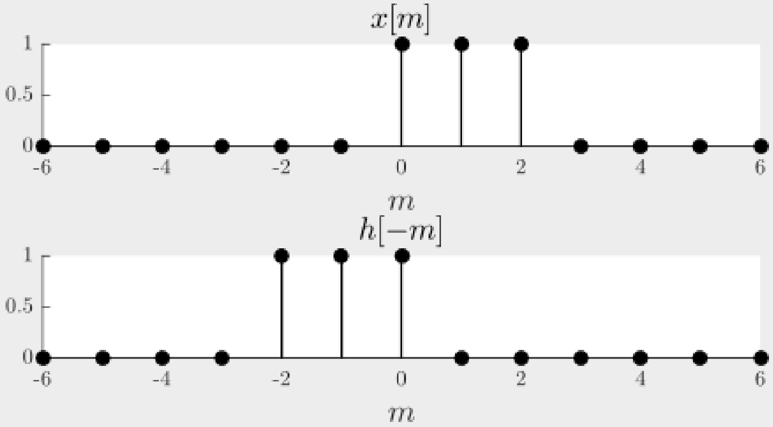
- Step 1: decide which of x[n] or h[n] you will flip and shift; you have a choice, since $x[n]*h[n]=h[n]*x[n]$. We will suppose you choose to flip h[n].
- Step 2: Plot x[m].
- Step 3: Plot $h[-m]$, the time-reversed version of the impulse response.
- Step 4: To compute y[n] at the time point n, plot the time-reversed impulse response after it has been shifted to the right (i.e, delayed) by n time units: $h[-(m-n)]=h[n-m]$.
- Step 5: y[n] is the inner product between the signals x[m] and $h[n-m]$ (technically it is the sum of the point-wise products; for complex signals, do not complex conjugate the second signal, as you would for a true inner product).
- Step 6: Repeat for all n of interest (potentially all $n \in Z$).
- Step 7: Plot y[n] and perform a reality check to make sure your answer seems reasonable.

Here we will illustrate convolution by convolving two signals together, x[n] and h[n], plotted below. We will call the result of the convolution y[n], so y[n] = x[n]*h[n].

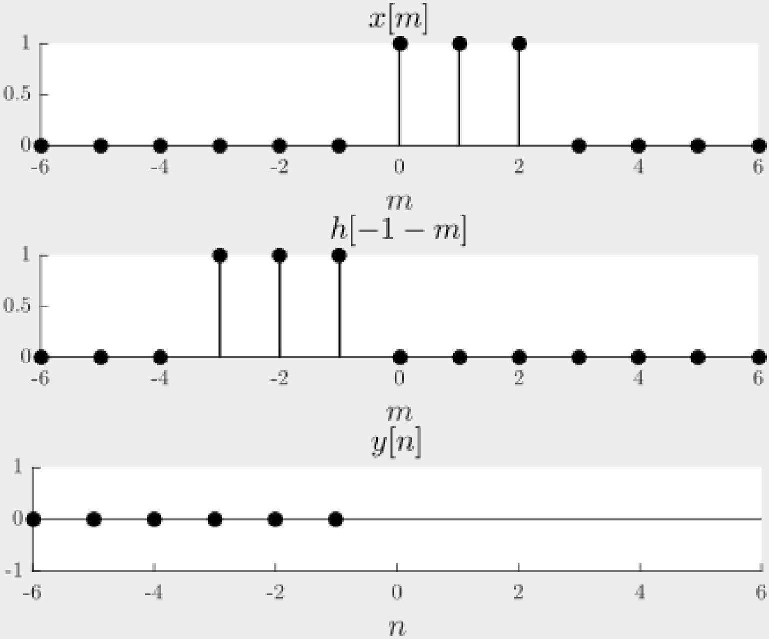


Recall that the first step of the convolution process is to decide which of the signals we will "flip and shift." The final result will be the same in either case, the only difference is in working the problem out. But here it truly does not matter, because the signals are identical. But we will say that we are flipping and shifting h[n].

The second and third steps are to plot x[m] and h[−m]:

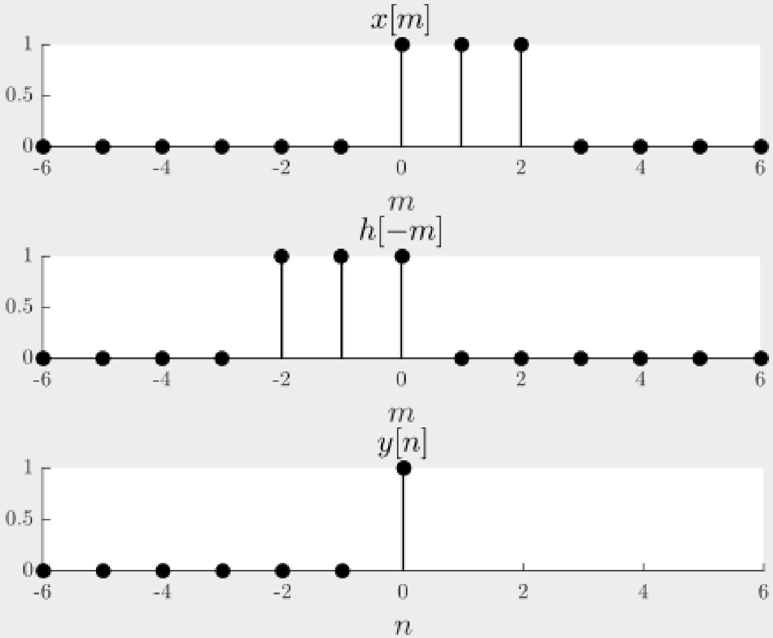


Now step 4 and 5 are to shift $h[n-m]$ by varying n and then computing the inner product between $h[n-m]$ and $x[m]$. For each shift, that inner product will be the value of the output $y[n]$ for that value of n. In the figures below, we will see how $h[n-m]$ and $x[m]$ look for each n ($x[m]$ of course will not change), and will progressively build up $y[n]$ as n increases. We'll start with $n=-1$:



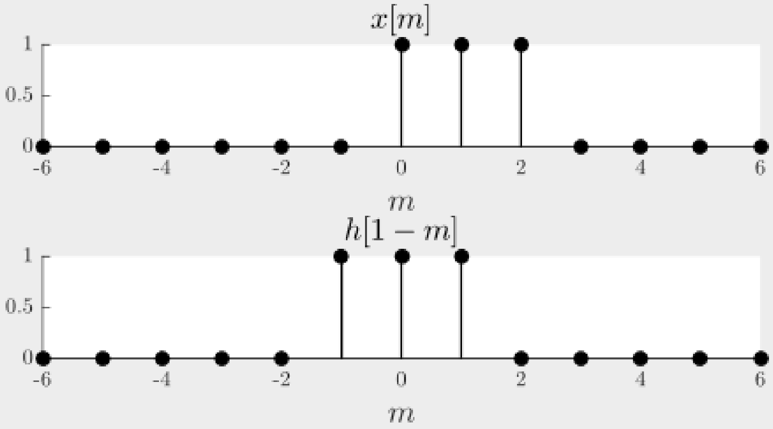
Notice how for this value of n, and for all n less than that as well (for which $h[n-m]$ will be even farther to the left), $x[m]$ and $h[n-m]$ have no overlapping nonzero values. Therefore the inner product in these cases is zero.

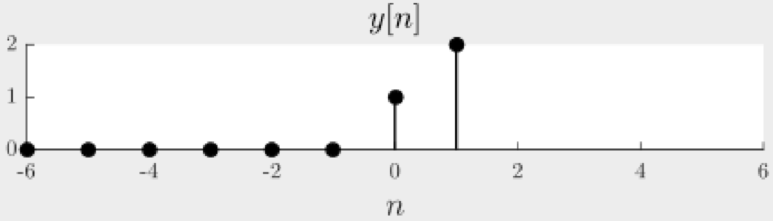
So for $n<0$, the dot products of $x[m]$ and $h[n-m]$ are zero, and hence $y[n]$ is zero for these n. Now let's consider $n=0$:



So now, for n = 0, x[m] and h[n − m] overlap just a bit (at m = 0), and the value of the dot product of the two signals is 1. So y[0] = 1.

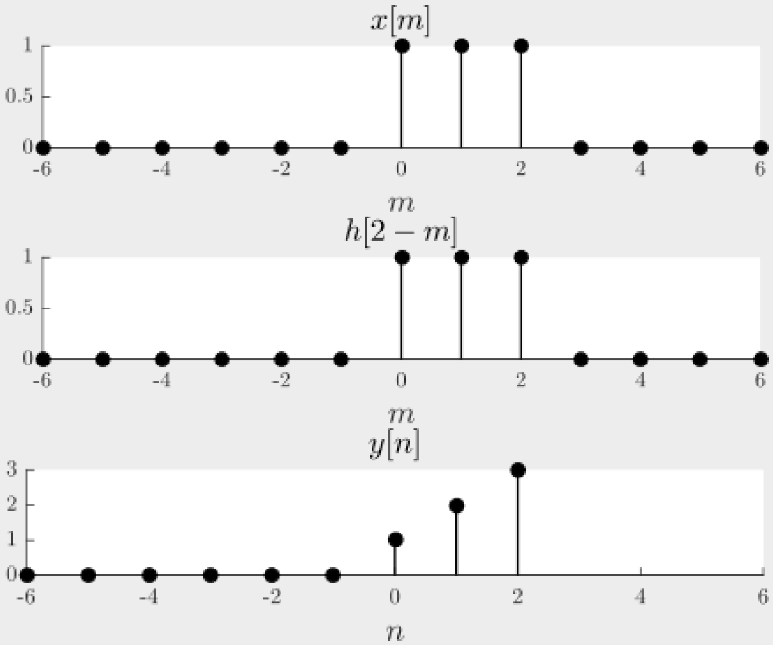
The next shift to the right makes n = 1:





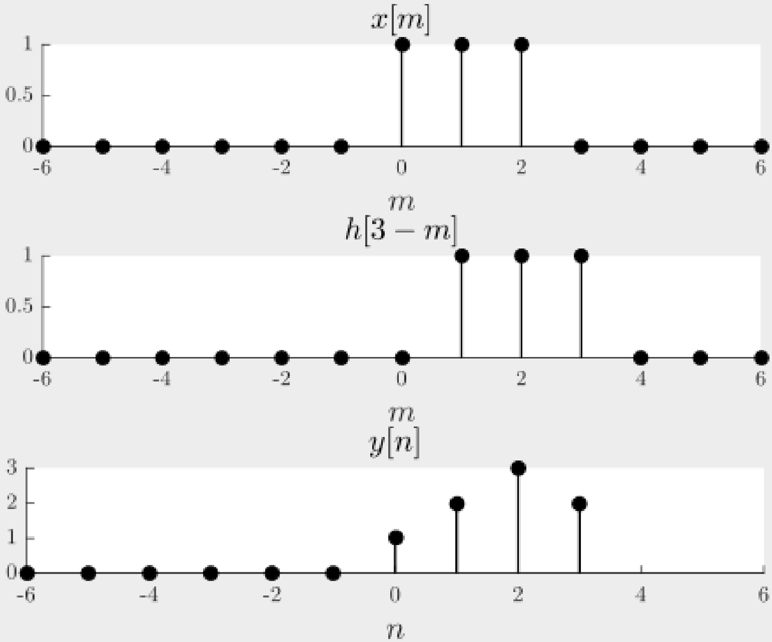
There are two nonzero locations where x[m] and $h[n-m]$ line up, $m=0$ and $m=1$. The dot product of the signals for this $n=1$ is $1{\cdot}1+1{\cdot}1=2$, so $y[2]=2$.

Continuing to shift $h[n-m]$ to the right, we have for $n=2$:

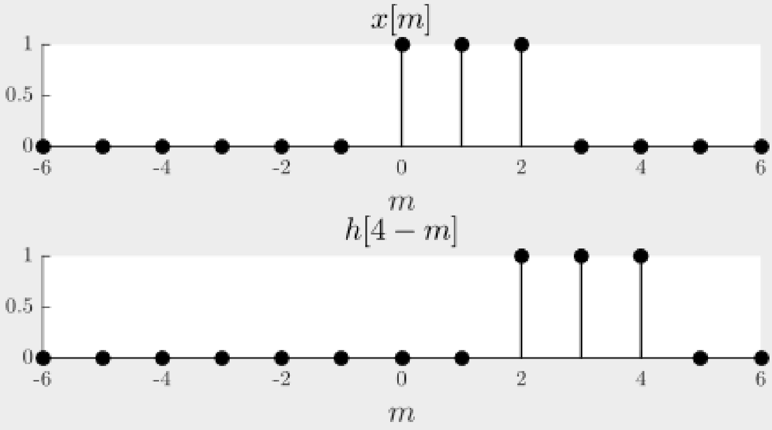


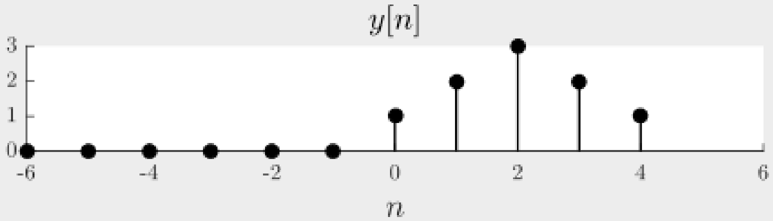
For $n=2$, the two signals are now maximally aligned, with a dot product of $1{\cdot}1+1{\cdot}1+1{\cdot}1=3$. So $y[2]=3$.

At $n=3$, $h[n-m]$ has moved past the point of maximum alignment; as with $n=1$, the dot product between it and x[m] is 2:

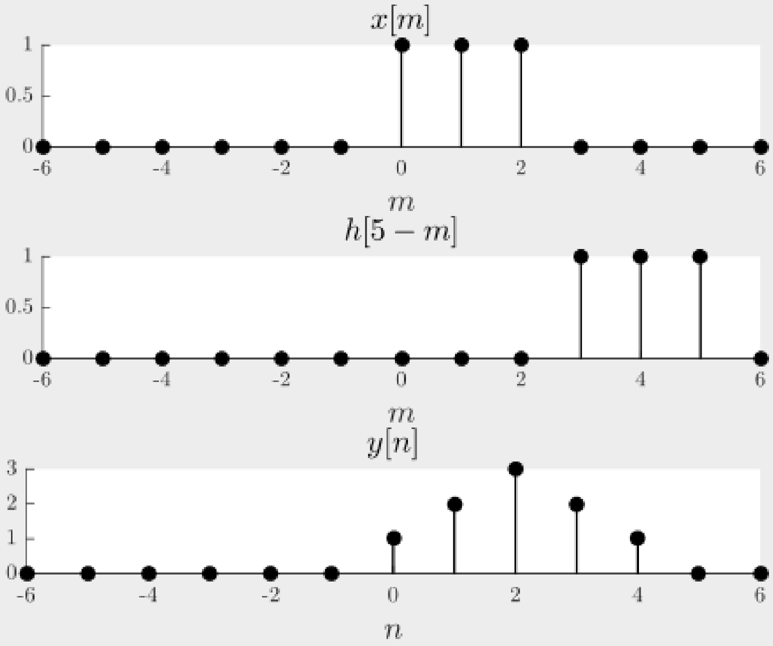


We're approaching the end of our work on this convolution problem. At n = 4, h[n − m] has just about moved on past the point of overlapping nonzero values with x[m]:





The dot product between the two signals is only 1. Finally, for n = 5, there is no nonzero overlap between x[m] and h[n − m], so the dot product is zero. For all n > 5 (for which h[n − m] moves even farther to the right) this is the case as well:



We have found the dot product between x[m] and h[n − m], and therefore y[n], for all integer values of n, so our convolution computation is finished. The final step of the convolution is to examine the plot of y[n] and do a quick "reality check" to make sure the answer is reasonable. At this introductory point, you might not know what to expect a convolution looks like, so it may not be clear if it is reasonable or not. But in time, especially as you understand the "flip and shift" nature of convolution, you will gain a feel for how convolutions should generally look.

Convolution of Finite-Length Discrete-Time Signals

Given a discrete-time, finite-length LTI system with an impulse response $h[n]$, it is a common operation to find the system's output $y[n]$ for some input $x[n]$. There are several ways this can be done, including finding the system matrix H and then using it to find the output via matrix multiplication: $y=Hx$. Recalling the circulant structure of that matrix, the output can be expressed in formula form: $y[n] = x[n] \circledast h[n] = \Sigma m=0N-1h[(n-m)N]x[m]$. This operation is known as **circular convolution** (or **finite-length convolution**).

Computing Circular Convolution

The circular convolution $x[n] \circledast h[n]$ can be computed using the following seven step procedure:

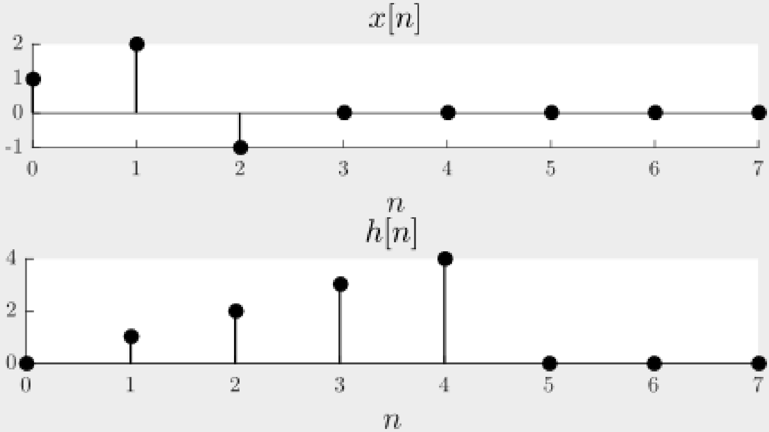
1. Decide which of $x[n]$ or $h[n]$ you will flip and shift. Since circular convolution is commutative, either choice will result in the same output.
2. Plot the values of $x[m]$ on a "wheel," beginning on the right side of it (for $m=0$), and continuing counter-clockwise for a total number of places that is equal to the length of the signal.
3. Plot $h[(-m)N]$ on another wheel. This simply

amounts to plotting the values in a clockwise direction. For a given value of n, y[n] will be the dot product of the two "wheels." Your original plots of x and h correspond to n=0. For n=1, first rotate the values of h in a counter-clockwise manner, i.e., delay them one time unit. Rotating h in this way is what creates h[(n−m)N]. Then...

4. Rotate the h wheel accordingly and...
5. Perform the inner product.
6. Repeat this for n ranging from 0 to N−1.
7. Plot your final answer and do a "reality check" to see if it seems reasonable.

Let's see the circular convolution process in action. Consider the two finite-length (N=8) signals below:

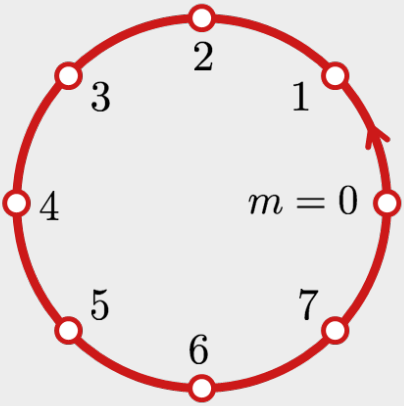
Discrete-time finite-length signals x[m] and h[m], N=8.



Since N=8, our "wheel" will have have 8 places.

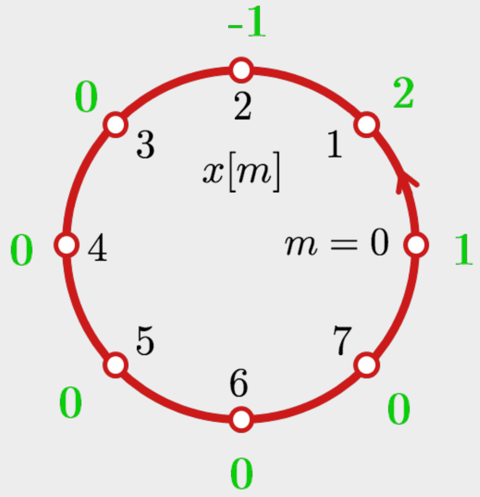
We will place the initial value on the right side of the wheel, but the starting place is arbitrary, so long as you are consistent (you can also choose to have your signal run clockwise, if you prefer).

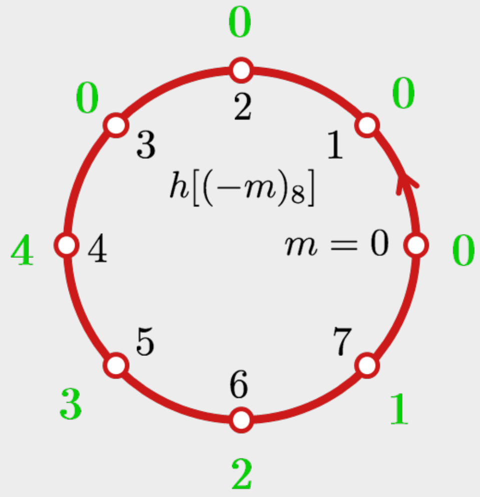
For the purposes of computing circular convolution by hand, the values of the signals may be plotted on a "wheel" with a number of places equal to the length of the signals. The time values progress counter-clockwise.



The second step is to plot x[m] and h[(−m)8]. For x, this means starting on the right and then writing the values counter-clockwise. For h, this means writing the values clockwise:

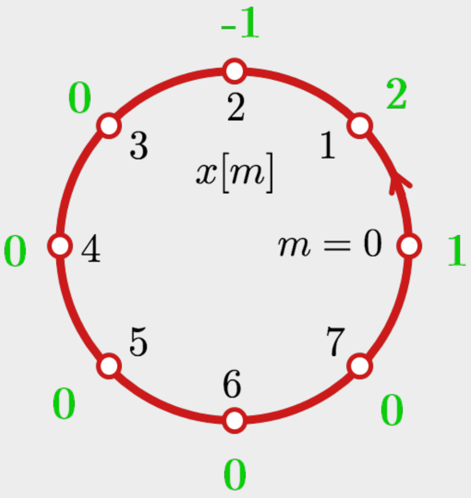
Plots of x[m] and h[(n−m)8]. The values of x[m] are plotted counter-clockwise. To plot h[(−m)8], the values of h[m] are simply plotted clockwise, i.e., backwards in time, starting at the same point.

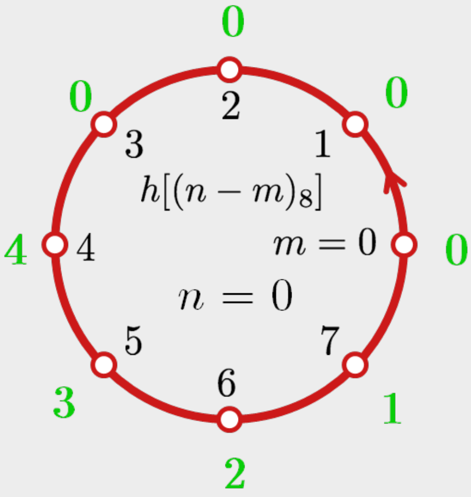




Having plotted x[m] and h[(−m)8], we can now begin to find our output values corresponding to each n. We'll start with n=0, which for h[(n−m)8] is of course what we have originally plotted:
Plots of x[m] and h[(0−m)8], the inner products of which is y[0]. The plot of x[m] will remain constant for each of the inner products. For the plot h[(n−m)8], the h[(−m)8] signal is rotated n

places units counter-clockwise. Here it is plotted for $n = 0$.





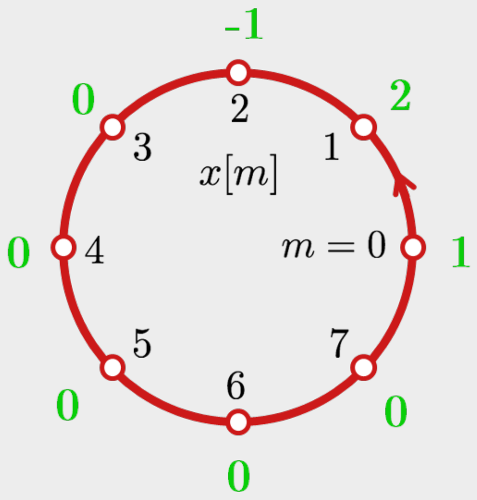
We'll move around counter-clockwise, starting at zero position, to find the inner product. The value of the inner product for $n = 0$ is:

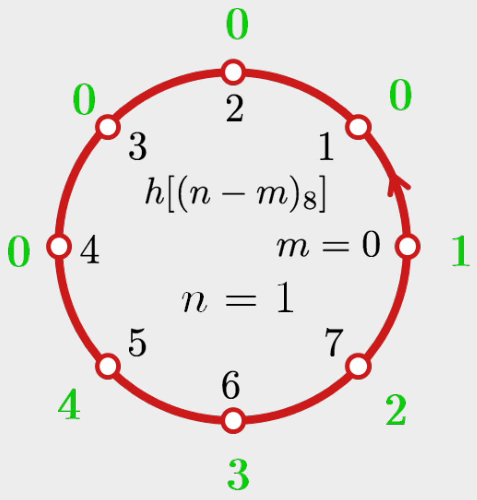
$y[0] = 1 \cdot 0 + 2 \cdot 0 +$
$-1 \cdot 0 + 0 \cdot 0 + 0 \cdot 4 + 0 \cdot 3 + 0 \cdot 2 + 0 \cdot 1 = 0.$

Now, for each successive value of n, starting with

$n=1$ and going up to $n=7$, we will rotate the h plot counter-clockwise one time unit, and then find the inner product with the (unchanged) x.

$x[m]$ and $h[(n-m)8]$ for $n=1$.



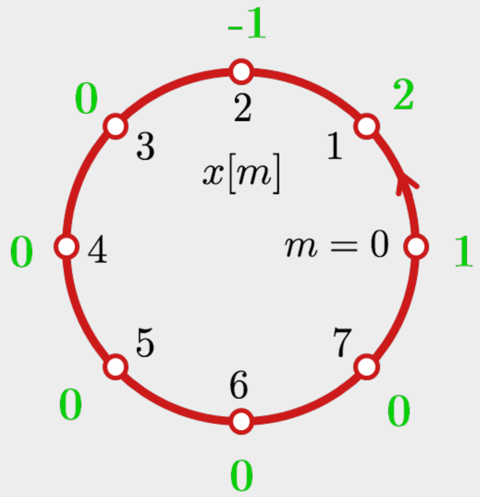


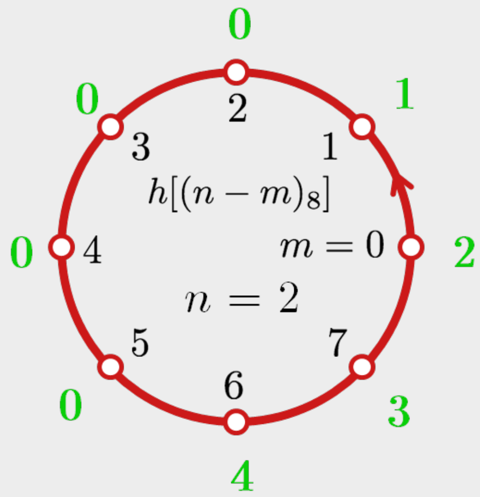
The value of the inner product for $n=1$ is:

$$y[1]=1\cdot 1+2\cdot 0+ -1\cdot 0+0\cdot 0+1\cdot 0+0\cdot 4+0\cdot 3+0\cdot 2=1.$$

Below are the plots for $n=2$. x remains unchanged, and h is shifted one time unit counter-clockwise from where it was at $n=1$:

x[m] and h[(n−m)8] for $n=2$.



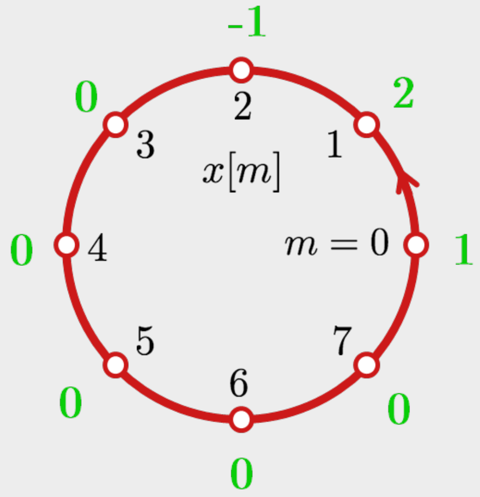


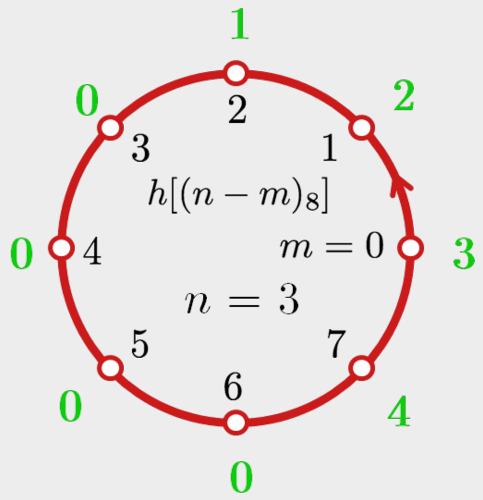
The value of the inner product for $n=2$ is:

$y[2]=1\cdot1+2\cdot0+-1\cdot0+0\cdot0+0\cdot0+0\cdot4+0\cdot3+0\cdot2=2+2=4$.

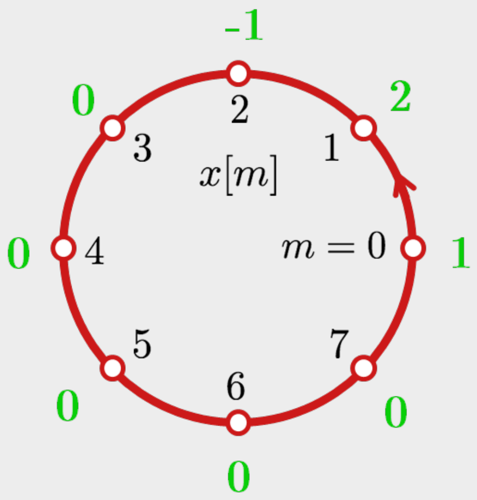
Now for $n=3$:

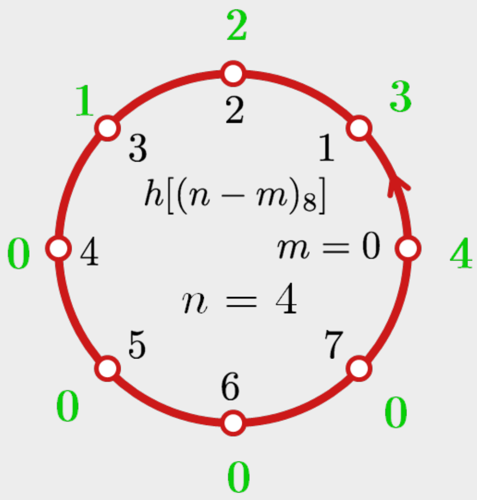
x[m] and h[(n−m)8] for $n=3$.



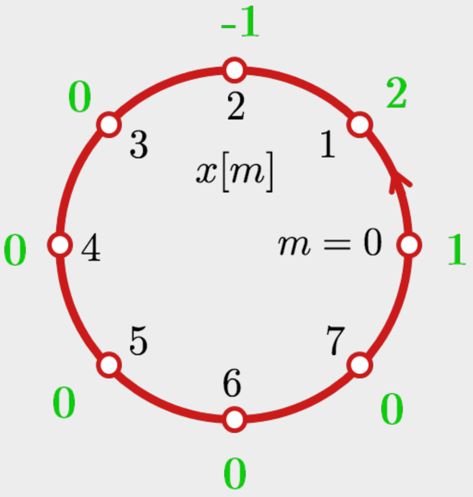


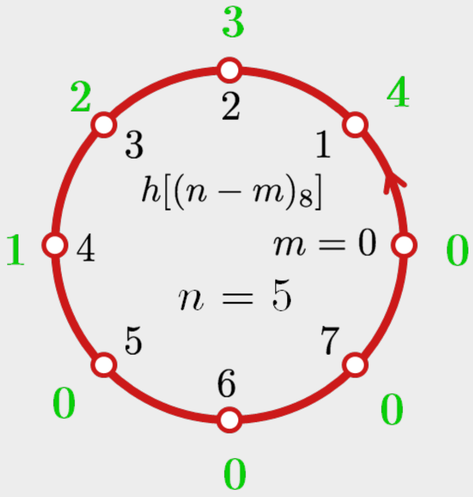
The value of the inner product for $n = 3$ is:
$y[3] = 1\cdot3 + 2\cdot2 +$
$-1\cdot1 + 0\cdot0 + 0\cdot0 + 0\cdot0 + 0\cdot0 + 0\cdot4 = 3 + 4 - 1 = 6.$
Next, $n = 4$:
$x[m]$ and $h[(n-m)8]$ for $n = 4$.



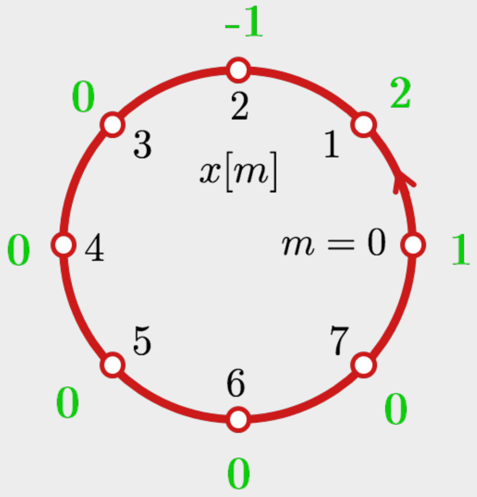


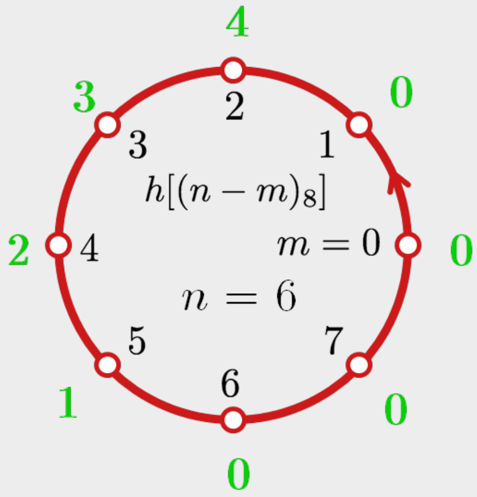
The value of the inner product for $n = 4$ is:
$y[4] = 1\cdot4 + 2\cdot3 + -1\cdot2 + 0\cdot1 + 0\cdot0 + 0\cdot0 + 0\cdot0 + 0\cdot0 = 4 + 6 - 2 = 8$.
For $n = 5$:
$x[m]$ and $h[(n-m)8]$ for $n = 5$.





The value of the inner product for n = 5 is:
$y[5] = 1{\cdot}0 + 2{\cdot}4 +$
$-1{\cdot}3 + 0{\cdot}2 + 0{\cdot}1 + 0{\cdot}0 + 0{\cdot}0 + 0{\cdot}0 = 8 - 3 = 5.$
For n = 6:
x[m] and h[(n − m)8] for n = 6.



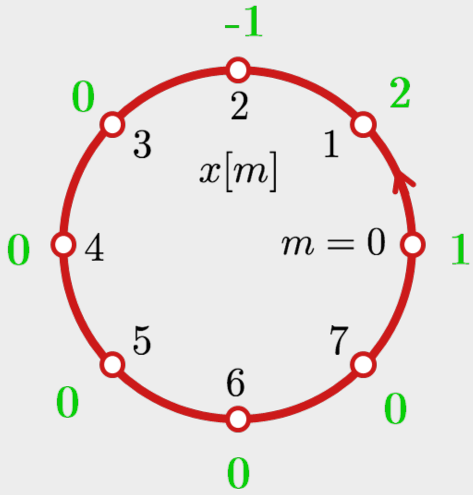


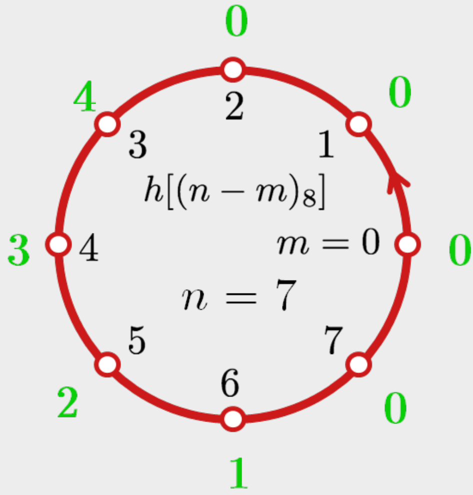
The value of the inner product for $n = 6$ is:

$y[6] = 1 \cdot 0 + 2 \cdot 0 + -1 \cdot 4 + 0 \cdot 3 + 0 \cdot 2 + 0 \cdot 1 + 0 \cdot 0 + 0 \cdot 0 = -4.$

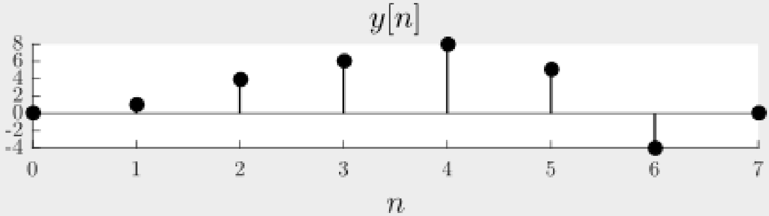
And finally, $n = 7$:

$x[m]$ and $h[(n-m)8]$ for $n = 7$.





The value of the inner product for n = 7 is:
$y[7] = 1 \cdot 0 + 2 \cdot 0 +$
$-1 \cdot 0 + 0 \cdot 4 + 0 \cdot 3 + 0 \cdot 2 + 0 \cdot 1 + 0 \cdot 0 = 0.$
Now that we have found the output for each n from 0 to 7, we may plot y[n].
The final plot of y[n].


$y[n]$
8
6
4
2
0
-2
-4
0
1
2
3
4
5
6
7
n

Properties of Discrete-Time Convolution

Recall the definition of discrete-time convolution. For infinite-length signals we have: $y[n] = x[n]*h[n] = \Sigma m=-\infty \infty h[n-m]x[m], -\infty<n<\infty$.

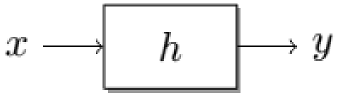
The formula for finite-length (with length N) signals is:

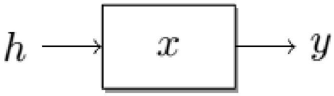
$y[n] = x[n]\circledast h[n] = \Sigma m=0N-1h[(n-m)N]x[m], 0\leq n\leq N-1$.

As with other mathematical operations, convolution has several important properties which we will now explore. We will express these properties in terms of infinite-length convolution; unless otherwise noted, they apply also to finite-time convolution.
 The commutativity property of convolution, as displayed in block diagram form.

Convolution is Commutative

Like addition and multiplication, the operation of convolution is commutative. For two signals x[n] and h[n], $x[n]*h[n]=h[n]*x[n]$. This means that when convolution is calculated by hand, either of the two signals can be the one chosen to "flip and shift." It also means that the output of a system with impulse response h[n] and input x[n] is the same as a system with impulse response x[n] and input h[n]. In block diagram form, these two diagrams are equivalent:



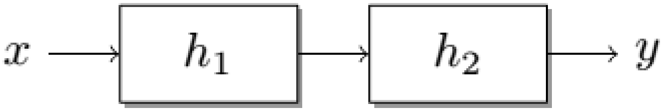


Though the output remains the same, appreciating the commutativity of convolution may give added insight to the convolution process.

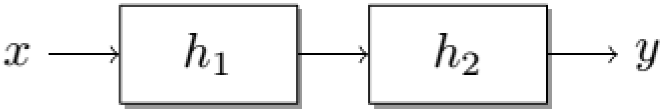
Proving the commutativity of convolution is straightforward; simply apply a change of variable to the convolution sum equation:
$y[n]=x[n]*h[n]=\Sigma m=-\infty\infty h[n-m]x[m] \text{Let } v=n-m \rightarrow m=n-v=\Sigma n-v=-\infty n-v=\infty h[v]x[n-v]=\Sigma v=\infty+nv=-\infty+nh[v]x[n-v]=\Sigma v=-\infty v=\infty h[v]x[n-v]=\Sigma v=-\infty v=\infty x[n-v]h[v]=h[n]*x[n]$
 A "cascade" of LTI systems. A cascade of two LTI systems is equivalent to a single LTI system with an impulse response that is the convolution of the cascaded systems' impulse responses.

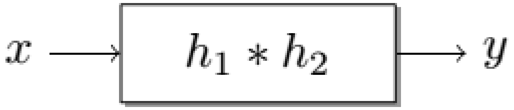
Convolution is Associative: Cascade of LTI Systems

Suppose a signal is processed by one LTI system, and the output is fed into another LTI system:



Can these two "cascaded" systems be simplified so that they are represented by a single system? It turns out that they can, because convolution is not only commutative, it is also associative: (x*h1)*h2=x*(h1*h2). Therefore if a signal x[n] is input into a system with impulse response h1[n], and the output of that system is then the input a system with impulse response h2[n], the final output y[n] is equivalent to that if x[n] had been input into a signal system with an impulse response of h1*h2:



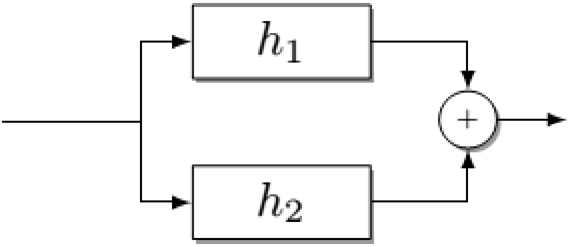


Proving the associativity of convolution is a matter of careful rearrangement of the sums: $(x*h1)*h2[n]=\sum_{m=-\infty}^{\infty} h2[n-m]\left(\sum_{l=-\infty}^{\infty} h1[m-l]x[l]\right)=\sum_{l=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} h2[n-m]h1[m-l]x[l]=\sum_{l=-\infty}^{\infty}\left(\sum_{m=-\infty}^{\infty} h2[n-m]h1[m-l]\right)x[l]=\sum_{l=-\infty}^{\infty}(h1*h2)[n-l]x[l]=x*(h1*h2)[n]$.

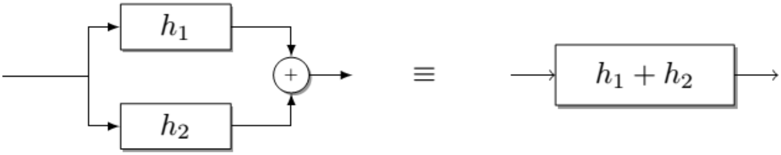
A parallel combination of LTI systems. A parallel combination of two LTI systems is equivalent to a single system whose impulse response is the sum of the parallel systems' impulse responses.

Convolution is Distributive: Parallel LTI Systems

Suppose now that one input x[n] is fed into two different LTI systems, and then the two outputs are then summed to form a new output:



Just as with the cascaded systems, this parallel arrangement of LTI systems can also be expressed as a single system, because of the distributive property of convolution, that $x[n]*h1[n]+x[n]*h2[n]=x[n]*(h1[n]+h2[n])$:



The proof of this property is a consequence of the distributive property of addition:$x[n]*h1[n]+x[n]*h2[n]=\Sigma m=-\infty\ \infty h1[n-m]x[m]+\Sigma m=-\infty\ \infty h2[n-m]x[m]=\Sigma m=-\infty\ \infty(h1[n-m]x[m]+h2[n-m]x[m])=\Sigma m=-\infty\ \infty(h1[n-m]+h2[n-m])x[m]=x[n]*(h1[n]+h2[n])$.

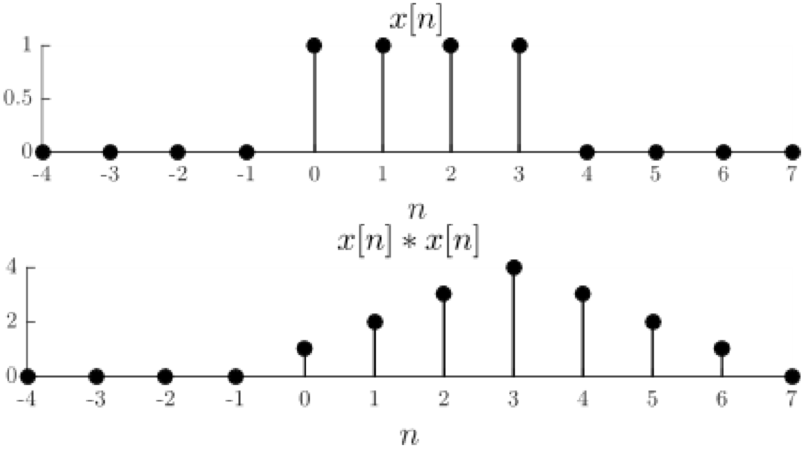
Discrete-Time Infinite-Length and Finite-Length Convolution Equivalence

When a signal x[n] of duration length of 4 is convolved with itself, the duration length of the result is $4+4-1=7$.

The Nonzero Support of the Convolution of Infinite-Length Signals

The convolution of two infinite-length signals yields another infinite-length signal. However, just because a signal is infinite in length does not mean it has an infinite number of nonzero values. The delta function δ[n] is nonzero only at $n=0$, even though it is infinite-length (i.e., it is defined for all integers n).

For reasons that will be apparent shortly, it is worth considering the **duration interval** of infinite length signals and their convolutions. Suppose an infinite-length signal is nonzero at all points less than $n=0$ and greater than $n=5$. Its duration interval goes from $n=0$ to $n=5$, and we say that the length of this duration is 6. It turns out that if signal a[n] has a duration length of Da, and signal b[n] has a duration length of Db, than the duration length of the convolution a[n]*b[n] is $Da+Db-1$.



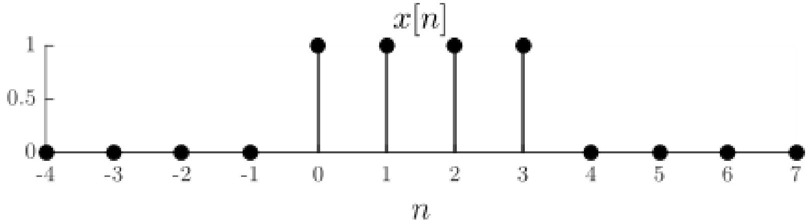
"Equivalence" of Linear and Circular Convolution

The reason why we would like to consider the duration length of two infinite-length signals' (linear) convolution is that there are situations in which we would like to produce the same result through circular convolution (i.e., the convolution of finite-length signals). Of course, the outputs of a linear convolution and circular convolution are not equivalent, strictly speaking; they produce two distinct categories of signals: the first one infinite-length, the second one finite-length. It is possible, however, to take the duration sections of infinite-length signals and treat them as finite-length signals *so that their (circular) convolution is the same as the duration section of the infinite (linear) convolution of the infinite-length signals*.

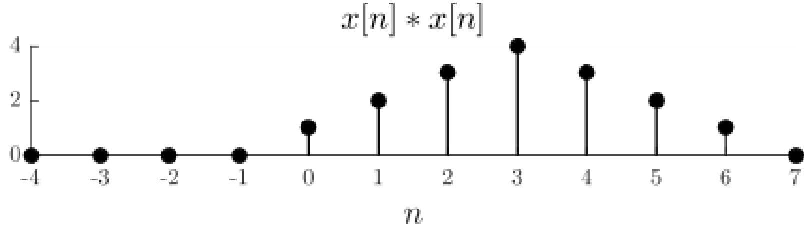
The finite-length signal x∼[n] here is simply the duration section of the infinite-length signal x[n] from above. The circular convolution x∼[n]⊛x∼[n] is clearly not the same as the duration section of the convolution of x[n] with itself. Once again, the circular convolution in this case is clearly not the same as the nonzero support of the convolution of x[n] with itself.

Zero-padding and Resulting Circular Convolutions

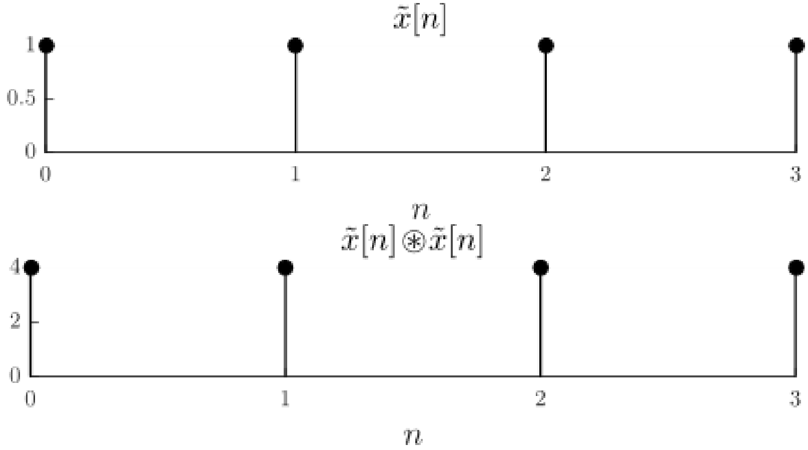
This is all perhaps best explained with illustrations. Consider the following infinite-length signal x[n] (obviously only a portion of it is shown):



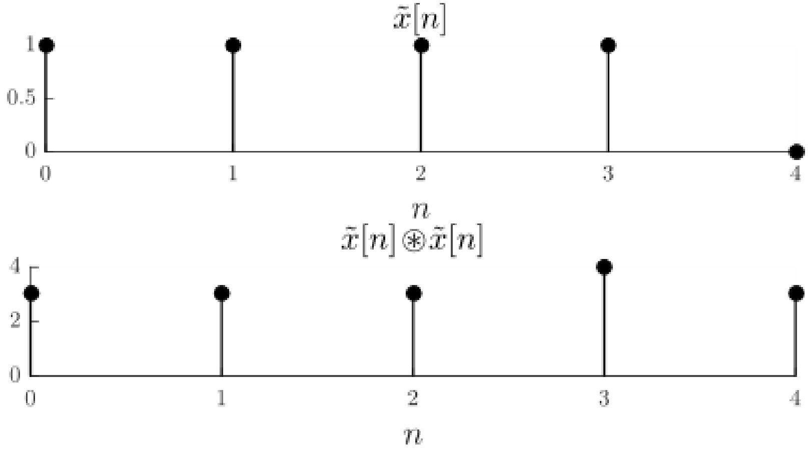
This convolution of this signal with itself produces another infinite-length signal (again, only a portion of it is shown):



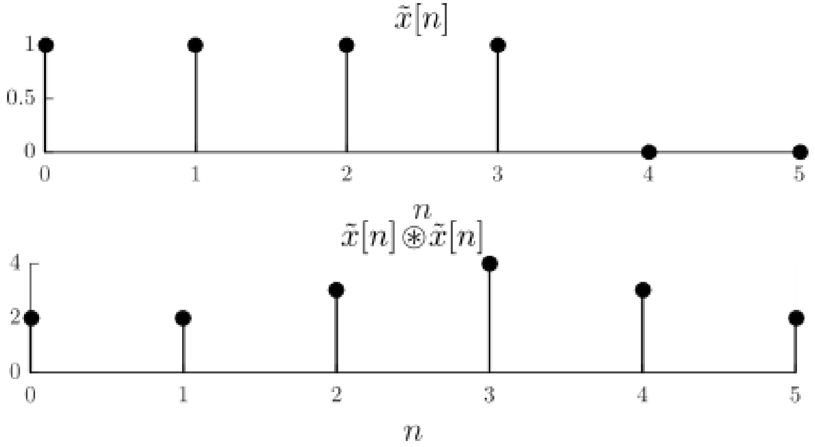
We will attempt to produce finite-length signal that is equal to the duration section of that convolution above (the nonzero portion, from n=0 to n=6). To do this, we will first try to circularly convolve two finite-length signals that are simply the duration section of x[n]. We will call this signal x~[n]:



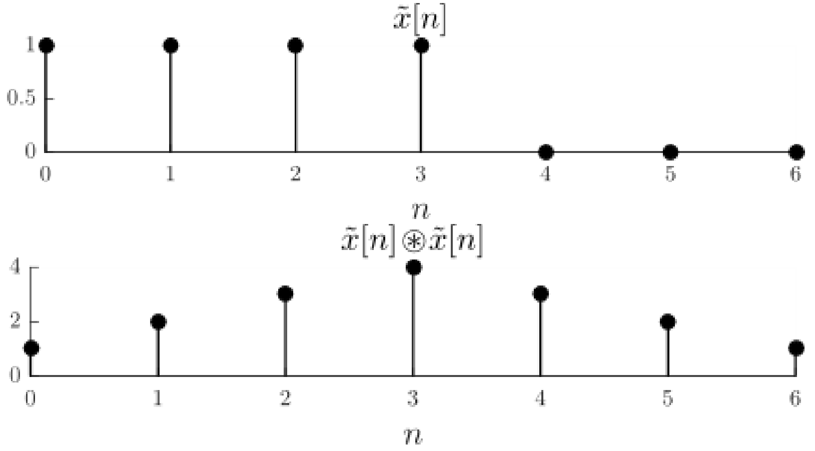
So it is apparent that simply circularly convolving the duration sections of x[n] will not do. We consider what would happen if we add a 0 to the end of x~[n] and try again.



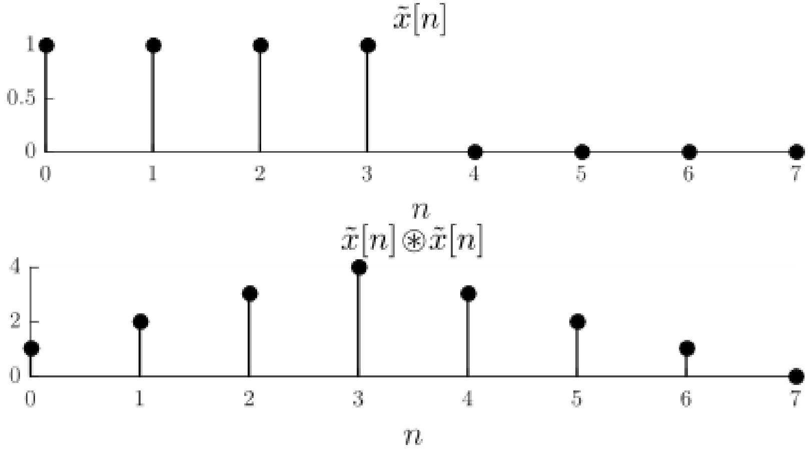
Well, this does seem to be a step in the right direction of having x∼[n]⊛x∼[n] equal the duration section of x[n]∗x[n]. Let's add another 0 to x∼[n]:



We are getting close now, and perhaps you have already figured out how many zeros we need to ultimately add to achieve the desired circular convolution. Let's add another:



So it seems that three zeros added to make x∼[n] did the trick, for the circular convolution x∼[n]⊛x∼[n] is indeed equal to the duration of x[n]*x[n]. Let's see, though, what happens if we add yet another zero:



At this point, it seems that adding extra zeroes to x~[n] will also add zeroes to the circular convolution; there's no harm with that, but it isn't necessary for our goal of the circular convolution equaling the duration section of the linear convolution.

Requirements of Circular/Linear Convolution Equivalence

Suppose we have two signals that produce some result via linear convolution; if we would like to achieve that same result (or at least, the duration section of that result) through circular convolution, then we will need to cut out the relevant sections of the two signals and then zero-pad them before circularly convolving them.

Let's define the amount of zero-padding with a bit more rigor. Consider infinite-length discrete-time signals a[n] and b[n]. Let anz[n] and bnz[n] be finite-length signals which are the duration sections of a[n] and b[n], respectively; that is, they are the smallest contiguous sections of a[n] and b[n] outside of which *all* values in time *before* and *after* those sections are zero (it is, of course, allowable that some values *within* the contiguous sections may be zero). Call the duration length of these duration sections Da and Db, respectively. Now zero-pad the signals anz[n] and bnz[n] (i.e., appended zeroes to the end of the signals) to bring the new length of these signals to be $Da+Db-1$, and call the new zero-padded signals $a{\sim}n$ and $b{\sim}n$. Then we have that $a{\sim}n \circledast b{\sim}n$ is equivalent to the duration section of $a[n]*b[n]$.

Put more simply, suppose that the duration length of $a[n]*b[n]$ is D (which we have seen equals $Da+Db-1$). We must zero-pad the duration sections of a[b] and b[n] to each be a length of D to have their circular convolution be equivalent to the duration section of the linear convolution.

Why this All Matters

There is a very good reason to go into this somewhat tedious task dealing with duration sections and their lengths and the relationships of linear and circular convolution. The reason has two parts. First: real-world input/output LTI systems implement *linear* convolution. Second: there are tremendous computational complexity advantages to performing *circular* convolution (which we will see later). So, if we can express the output of linear convolution through circular convolution, it means we can perform real-world system computations quickly.

Impulse Response and LTI System Causality

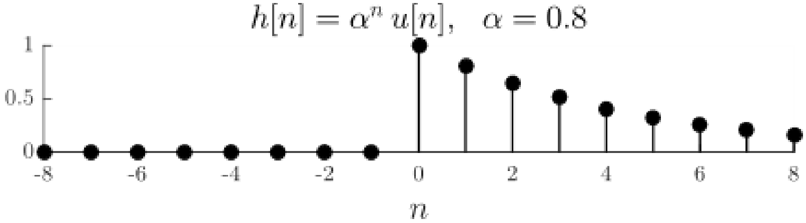
In addition to linearity and time-invariance, there are other significant classifications of discrete-time systems. One of these is causality. A system is **causal** if its output, for any n, depends only on inputs at or before time n. Causality is an important practical qualification on a system: it is not possible to implement a real-time system whose current output depends on future values! While the qualification of causality is also relevant for non-LTI systems, it has a special relationship with LTI systems. Recall that for LTI systems, the impulse response can be used to find the system's output given some input (through convolution of the input and the impulse response). Likewise, for LTI systems, the impulse response can also tell us whether or not the system is causal.

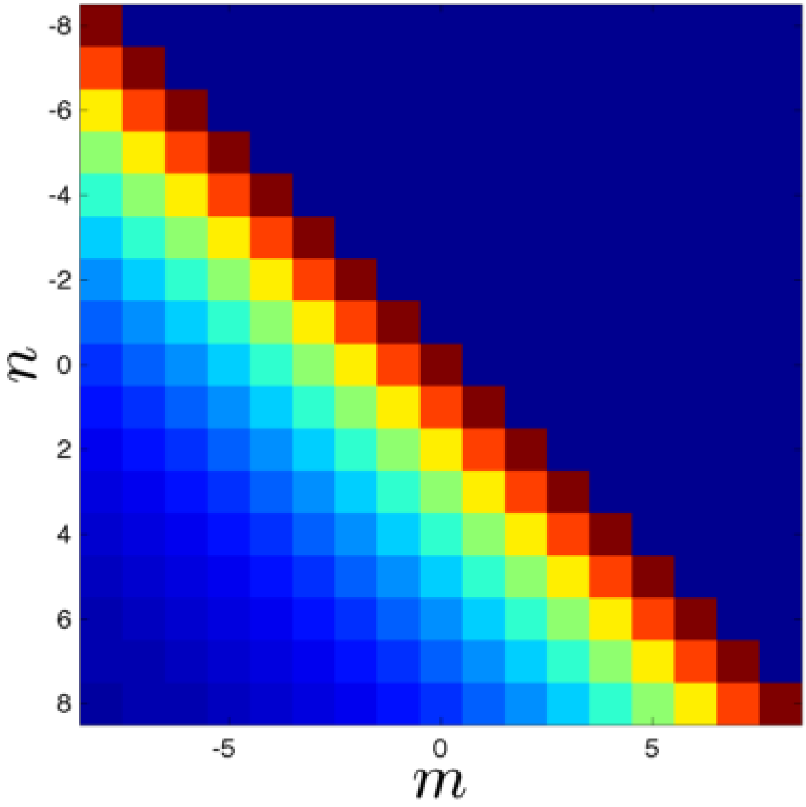
An example of an impulse response for a causal system. The impulse response is zero for $n<0$.

The Impulse Response and Causality

There is a straightforward relationship between an LTI system's impulse response and whether or not the system is causal: An LTI system is **causal** if and only if its impulse response is 0 for all $n<0$ (i.e., the impulse response is a causal signal). This follows naturally from the convolution sum. The system's output is defined as $y[n]=\Sigma m=-\infty\infty h[n$

$-m]x[m]$. Note that if and only if $h[m] = 0$, $\forall\ m < 0$, no $x[m]$ for $m > n$ contribute to the sum, which is to say that no future values of the input contribute to the output at time n. Below is an example of the impulse response for a causal system. Note that it is 0 for $n < 0$. This also corresponds to the system matrix being lower triangular:

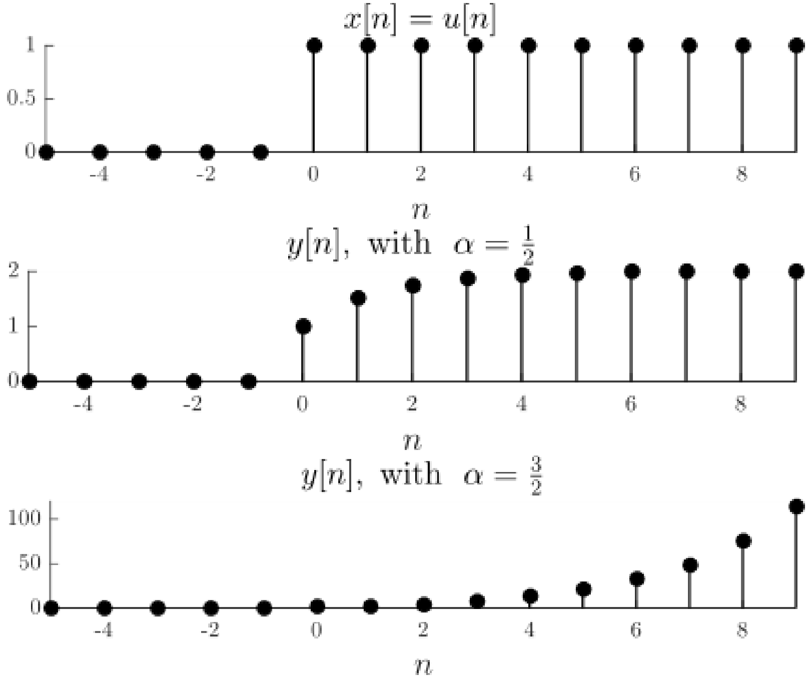




Impulse Response and LTI System Stability

It is of practical significance in the design of discrete-time systems that they be "well behaved," meaning that for any "well behaved" input, the system gives a "well behaved" output. Colloquially speaking, we do not want an innocuous input to result in the system "blowing up." The technical term for "well behaved" systems is that they are **stable**. It is essential for many systems that they be stable, for the sake of safety and proper operation within wider systems. For example, with steering or braking or aircraft control systems, it could be catastrophic if a small input led to a wildly divergent output.

Consider the recursive average system $y[n] = x[n] + \alpha y[n-1]$, with an eminently reasonable and contained input of the step function $u[n]$. For values of α less than 1, the system is "well behaved," but the output "blows up" for $\alpha > 1$: When the input to a recursive average system is a step function, the output may or may not be contained, depending on the value of the α coefficient. For $\alpha < 1$, the system's output is contained, whereas for $\alpha > 1$, the system's output increases with time exponentially.



Now as with any desirable characteristic or quality, there are many ways to define stability. We could say a system is stable if its output never exceeds a particular value, or perhaps that the output's energy per some time period is capped. One of the most common ways to define stability is **bounded-input bounded-output (BIBO) stability.**

A system is said to be BIBO stable if, for any bounded input (meaning that the magnitude of the signal never exceeds some finite value), the output is also bounded (but not necessarily by the same value as the input). Mathematically, we can put it like this:

Consider a discrete-time system H and arbitrary input signal x[n] with $|x[n]| \leq M1 \in R$, $\forall n$. Let $y[n] = H\{x[n]\}$. H is BIBO stable if there exists some $M2 \in R$ such that $|y[n]| \leq M2$, $\forall$ n.

BIBO stability is a guarantee, a stamp of approval on a discrete-time system, certifying that the output will *always* be capped by some value...so long as the input also is.

The Impulse Response and BIBO Stability in LTI Systems

For LTI systems, there is again a special relationship between the system and its impulse response. The system's impulse response can tell us two things regarding the system: if it is BIBO stable, and if so, the particular value that bounds the output.

Consider a discrete-time LTI system H with impulse response h[n], and arbitrary input signal x[n] with $|x[n]| \leq M1$, $\forall$ M. H is BIBO stable if and only if there exists some M2 such that $\|h[n]\|1 = M2$. Furthermore, this M2 (if it exists) bounds the output of the system: $|H\{x[n]\}| \leq M1M2$.

Proof

The proof of our "if and only if" statement has two parts. First, we must show that the existence of $\|h[n]\|_1$ implies BIBO stabilty. Second, we must show that BIBO stability implies the existence of $\|h[n]\|_1$. We'll start with the first part, the "if." Let H be a system with bounded impulse response $\|h[n]\|_1 = M_2$. Consider the output corresponding to an arbitrary input x[n] which is bounded by M_1:

$$|y[n]| = |H\{x[n]\}| = \left|\sum_{m=-\infty}^{\infty} h[n-m]x[m]\right| = \left|\sum_{m=-\infty}^{\infty} x[n-m]h[m]\right| \le \sum_{m=-\infty}^{\infty} |x[n-m]||h[m]| \le \sum_{m=-\infty}^{\infty} M_1|h[m]| \le M_1\sum_{m=-\infty}^{\infty} |h[m]| \le M_1M_2.$$

The output is bounded by a finite value, M_1M_2. So, if the impulse response $\|h[n]\|_1$ for an LTI system exists, then the system is BIBO stable.

The "only if" side of the proof is to show that if a system is BIBO stable, the norm $\|h[n]\|_1$ of its impulse response must exist. We will demonstrate this by proving the contrapositive: if $\|h[n]\|_1$ is unbounded, the system is not BIBO stable. It will require the use of a the function sgn{x[n]}, which outputs 1 when x[n] is positive, -1 when it is negative, and 0 when it is 0.

Consider an arbitrary impulse response h[n] that is not absolutely summable, i.e., $\|h[n]\|_1$ is unbounded. For the system to be BIBO stable, any bounded input must result in a bounded output. So let the input

$x[n]=\mathrm{sgn}\{h[-n]\}$. Clearly, $x[n]$ is bounded: $\|x[n]\|\infty=1$. But

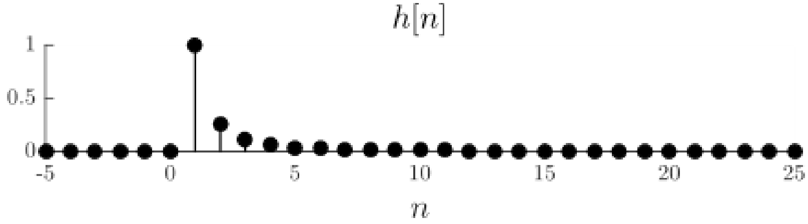
$$y[0]=\Sigma m=-\infty\infty h[0-m]x[m]=\Sigma m=-\infty\infty h[0-m]\mathrm{sgn}\{h[m]\}=\Sigma m=-\infty\infty|h[m]|=\|h[n]\|1.$$

So in this case $y[0]$ is unbounded, so the system is not BIBO stable.

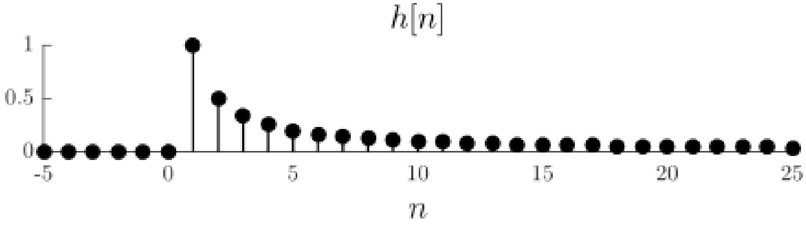
Example of an impulse response of a BIBO system. Example of an impulse response of a system that is not BIBO stable.

Examples of BIBO and non-BIBO Systems

Suppose an LTI system (remember, impulse response absolute summability being equivalent to BIBO stability *only applies to LTI systems!*) has the impulse response $h[n]=1n2u[n-1]$:



As the norm $\|h[n]\|1$ of this impulse response exists ($\Sigma n=1\infty|1n2|=\pi 26$), the system is BIBO stable. But suppose $h[n]=1nu[n-1]$:



For this system, the norm $\|h[n]\|1$ does not exist, as $\Sigma n=1\infty|1n|\rightarrow\infty$. So this system is not BIBO stable.

Because of this impulse response property, it is evident that all FIR systems are BIBO stable (for a finite sum of finite values is finite).

Conclusion

Once again we have seen the significance of the impulse response for LTI systems. They can be used to compute a system's output, and they can be used to determine the causality of a system. Here we have seen they can also be used to determine the BIBO stability of a system, a characteristic of great importance. But note two things. First, the connection between the impulse response and stability (as with those other connections) applies only to LTI systems. For example, the system $H\{x[n]\}=1x[n]+1-1$ is not stable (the output explodes as the input tends to -1), yet its impulse response is $-12\delta[n]$, which is absolutely summable. And second, just because a system is not BIBO stable, this does not necessarily mean that it is not useful, but rather that care must be given to what is input into the system.

Orthogonal Bases

Recall from before, how we can understand discrete-time signals to be vectors in a vector space. There are some very useful reasons why we might want to express some signal in a vector space in terms of other signals in that space. To better understand how all of this works, and to give us some mathematical foundations for it, we will consider the concept of bases.

The Basis of a Vector Space

Suppose we have some vector space V, such as R^N or C^N, i.e., the set of all real or complex valued finite-length (of length N) discrete time signals. We define a **basis** for V as a set of vectors $\{b_k\}, b_k \in V$ which span V and are linearly independent. By spanning V, we mean that any vector in V can be expressed as a linear combination of one or more vectors in $\{b_k\}_{k=0}^{N-1}$:

$$\forall x \in V, x = \sum_{k=0}^{N-1} \alpha_k b_k = \alpha_0 b_0 + \alpha_1 b_1 + \cdots + \alpha_{N-1} b_{N-1}, \alpha_k \in C.$$

By the vectors in the set $\{b_k\}$ being linearly independent, we mean that no vector in that set can be expressed as a linear combination of any of the others. The number of these spanning and linearly independent vectors in the basis is the **dimension** of the basis. Our example bases RN or CN are of dimension--you guessed it--N.

The Basis Matrix

For the sake of cleaner and simpler mathematical expression--as well as the ability to connect the concept of a basis with other linear algebra tools--we can create a matrix with basis vectors as columns. If the dimension of the basis is N, then this collection of basis vectors will be an $N \times N$ matrix, which we'll call B:

$B \ = \ [b0|b1|\cdots|bN-1].$

Recall we can express any vector in a vector space as a linear combination of the basis vectors. We can put these weights $\{\alpha k\}$ into an $N \times 1$ column vector:

$a = [\alpha 0 \alpha 1 \vdots \alpha N-1].$

With the basis matrix B and the weights vector a, we can express the linear combination using a simple matrix multiplication: $x = Ba$. So we see that we can use the basis matrix and weights vector to refer to any vector in the vector space, through the linear combination of the basis vectors (which we can express with a matrix multiplication). Now, it is a natural question to ask, given some vector x in the vector space, how do we find the weights a that will produce the expression $x = Ba$? If we would like to express x as a linear combination of basis vectors, it is important that we know how to find those weights! Thankfully, it is very straightforward. Simply multiply each side of the matrix multiplication equation by the inverse of the basis matrix:

$$x = BaB - 1x = B - 1BaB - 1x = IaB - 1x = a.$$

The weight vector is simply the inverse of the basis matrix, times the vector x (the one we want to express in terms of the basis matrix).

Orthogonal and Orthonormal: Special Bases

We have seen that a basis is a special collection of vectors from some vector space: it is a collection that spans the space, and is mutually linearly independent. If we add another requirement or two, we end up with two important sub-classes of bases. Suppose the vectors in some basis are not only spanning and linearly independent (which of course they must be, by definition, to form a basis), but that they are also mutually orthogonal, that is, that the inner product of any basis vector with any other basis vector is 0:

$\forall\ bk \in \{bk\}k=0N-1, \langle bk,bl \rangle \ = \ 0, k \neq l.$

If such is the case, then this basis is said to be an **orthogonal basis**.

So an orthogonal basis is a particular kind of basis, one whose vectors are mutually orthogonal. Among orthogonal bases, there are some whose vectors all have unit 2-norms:

$\langle bk,bl \rangle \ = \ 0, k \neq l \| bk \| 2 \ = \ 1.$

A basis with this additional property is known as an **orthonormal basis**.

Basis Matrix of an Orthonormal Basis

Like any other basis, the vectors of an orthonormal basis can be put together to form a basis matrix. Recall how we find the weights a to express some vector x in terms of the basis matrix B: $a = B-1x$. The reason that orthonormal bases are so special is that, in contrast to other bases, their matrix inverses are extremely easy to find. For orthonormal basis matrices, $B-1 = BH$. The inverse of an orthonormal basis matrix is simply its Hermitian (conjugate) transpose! So for these bases, $a = BHx$. In linear algebra, a matrix that has the property that its inverse is simply its Hermitian transpose is called a **unitary matrix**. If the matrix is real-valued, it is called an **orthogonal matrix** (this is a bit of unfortunate nomenclature, considering its columns are actually mutually *orthonormal*), and its Hermitian transpose is simply the regular matrix transpose, B^T (sometimes written as B').

Orthonormal Basis Signal Representation

Putting this all together, we can see the two key aspects of signal representation with orthonormal bases. There is the **synthesis** side, that we can build up any vector x in the vector space through a linear combination of basis vectors. And there is the **analysis** side, that we can find the proper weights of this linear combination by simply multiplying x by the Hermitian transpose of the basis matrix.

- Synthesis: $x = Ba = \Sigma_{k=0}^{N-1} \alpha_k b_k$
- Analysis: $a = B^H x$,or, $\alpha_k = \langle x, b_k \rangle$

Eigenanalysis of LTI Systems (Finite-Length Signals)

In the study of discrete-time signals and systems, concepts from linear algebra often provide additional insight. When it comes to LTI systems, a certain area of linear algebra is particularly helpful: eigenanalysis.

Eigenvectors and Eigenvalues

Given a square matrix (one that has the same number of rows as columns) A, a vector v is an eigenvector with corresponding scalar eigenvalue λ if: $Av=\lambda v$. There is a geometric interpretation to this eigenanalysis of the matrix A. Multiplying a matrix by one of its eigenvectors produces simply a scaled version of that same eigenvector (scaled by a factor of λ), so a matrix multiplication of an eigenvector does not change its orientation, only its strength.

Consider this example in two dimensions, a square matrix A:

$A=[3113]$,

and the vector v:

$v=[1-1]$.

Note what happens when we multiply the matrix A by the vector v:

$Av = [3113][1-1] = [(3-1)$
$(1-3)] = [2-2] = 2[1-1] = 2v.$

So running the vector v through the matrix A simply scales the vector by 2.

Eigendecomposition: Handling Multiple Eigenvectors and Eigenvalues

We have seen what eigvenvectors and eigenvalues are for square matrices. Now, an $N \times N$ matrix will have N eigenvectors (not necessarily distinct), each with its own eigenvalue. We can put all of these vectors and values into their own matrices. Suppose the eigenvectors of the matrix are $\{vm\}m=0N-1$ and the values are $\{\lambda m\}m=0N-1$. Then we can organize all of them like this:

$V = [v0|v1|\cdots|vN-1]\Lambda = [\lambda 0\lambda 1\ddots\lambda N-1]$.

With those vectors and values collected like that, we can express the eigenvector/value property $Av = \lambda v$ for all of the eigenvectors and eigvenvalues of the matrix A at once: $AV = V\Lambda$.

Diagonalization

Now, if the square matrix V is invertible (which will be the case if the columns of A are linearly independent, which of course happens if the vectors form a basis), then we can do some special things with it.

Recall the eigendecomposition relationship: $AV=V\Lambda$. If V is invertible, then we can multiply each side of the equation by $V-1$: $V-1AV=V-1V\Lambda=I\Lambda=\Lambda$. So $V-1AV=\Lambda$. Because multiplying A by V on one side and $V-1$ on the other produces a diagonal matrix, we say that the matrix V ***diagonalizes*** A. If we multiply the eigendecomposition matrix the other way, we will have that $A=V\Lambda V-1$. One of the reasons (more of which we'll see later) why we consider this diagonalization of A is that it is easier to matrix multiply with diagonal matrices than full ones.

When a complex harmonic sinusoid is input into an LTI system, the output is a scaled version of the input. Here the real (cosine) and imaginary (sine) parts of the sinusoid are plotted as the input. Note how the system merely scales the inputs.

LTI Systems and Eigenanalysis

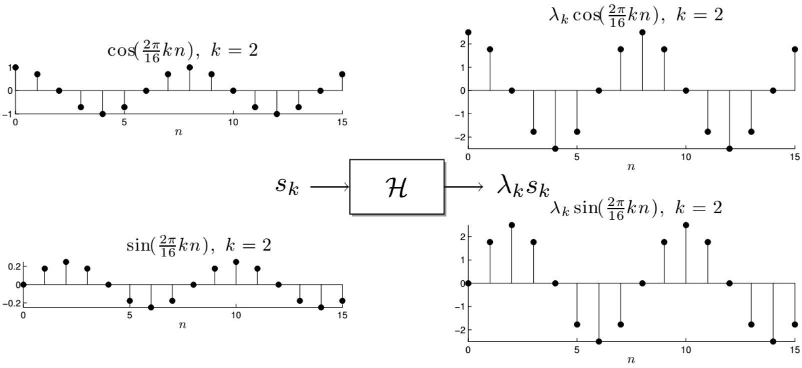
Perhaps you may be wondering how all of this linear algebra relates to discrete-time systems. For LTI systems operating on finite-length discrete-time systems, the input output relationship is: $y = Hx$, where H is a circulant matrix (each row being a circularly shifted version of the system impulse response h).

Let's see what the eigenvectors of H are. These will be the finite-length signals that, when input into the system, emerge as outputs simply as scaled versions of themselves. In that sense, they are somehow fundamentally related to all LTI systems.

It so happens that, remarkably, any and all LTI systems for finite-length (length N) signals have the exact same set of eigenvectors! The eigenvectors for any LTI length-N system are complex harmonic sinusoids:

$sk[n] = ej2\pi NknN = 1N(\cos(2\pi Nkn) + j\sin(2\pi Nkn)), 0 \leq n,$
$-1.$

So, if we have an LTI system--any LTI system--then giving an sk as an input will result in the output being λksk, with the particular values of λk of course being dependent on the system:



To prove this special property of LTI systems, we simply compute the circular convolution sum for an LTI system with arbitrary impulse response h[n] and input of the form ej2πNknN:

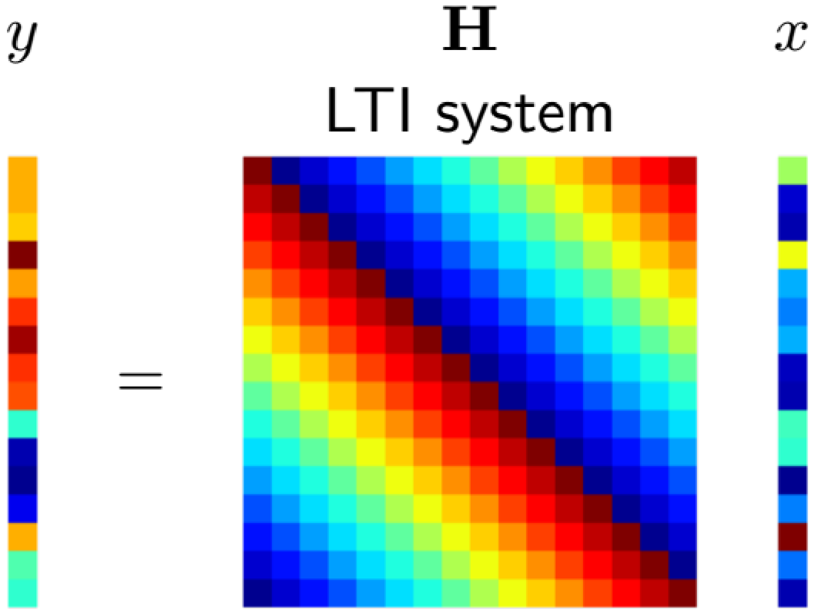
sk[n]⊛h[n] = Σm = 0N − 1sk[(n −m)N]h[m] = Σm = 0N − 1ej2πNk(n −m)NNh[m] = Σm = 0N − 1ej2πNk(n −m)Nh[m] = Σm = 0N − 1ej2πNknNe −j2πNkmh[m] = (Σm = 0N − 1e −j2πNkmh[m])ej2πNknN = λksk[n] , λk = (Σm = 0N −1e − j2πNkmh[m]).

This proof reveals how we are to find the eigenvalues (λk) that correspond to each harmonic sinusoid eigenvector: they are simply the inner products of the eigenvectors with the system's impulse response. Each value λk is called the system's *frequency response* at frequency k, because it indicates how the system scales inputs of that particular frequency. It is a significant enough characteristic of the system to warrant its own notation: H[k].

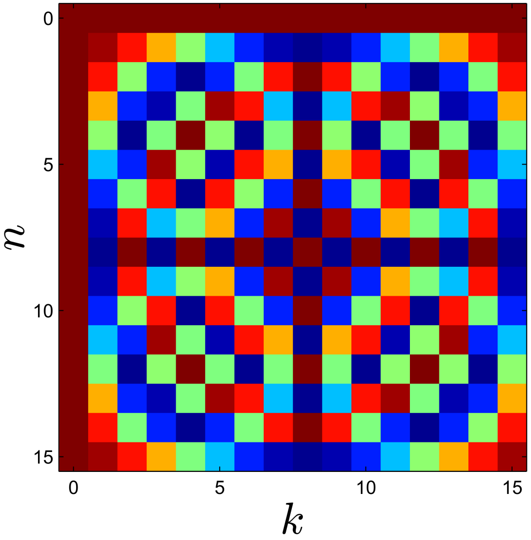
Graphical representation of the real and imaginary parts of the discrete-time finite length LTI system eigenvector matrix S.The real part of the eigenvector matrix S: cos(2πNkn)/N. The real part of the eigenvector matrix S: sin(2πNkn)/N. Eigendecomposition of discrete-time finite length LTI systems.

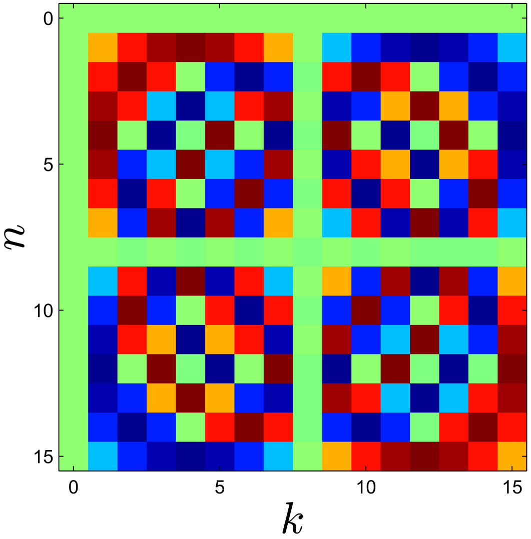
Eigendecomposition of LTI Systems

As with matrices in general, we can apply an eigendecomposition on an LTI system matrix. For LTI finite-length systems, these matrices are circulant:



We have seen that the eigenvectors of LTI systems are harmonic complex sinusoids. We can stack these up into a single matrix S, the entries of which are Sn,k, which is plotted below:

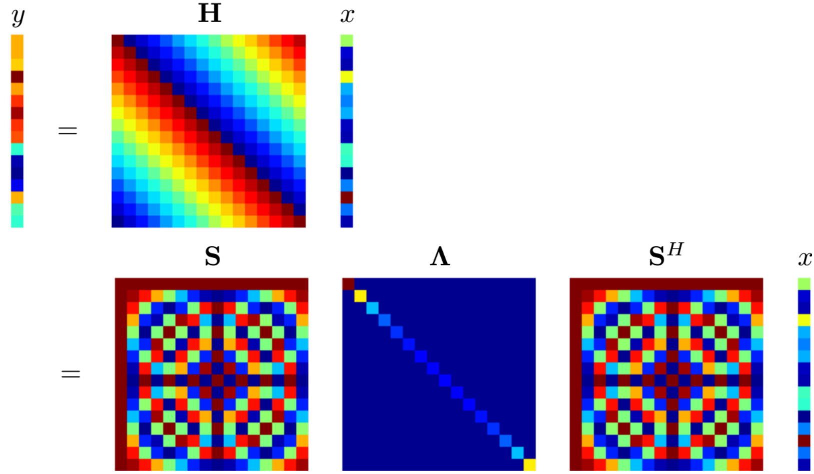




Likewise, we can plot the respective eigenvalues of the eigenvectors, which above we defined to be the values of the frequency response of the system:

$$\Lambda = [\lambda 0 \lambda 1 \dot{\cdot}. \lambda N - 1] \ = \ [Hu[0] Hu[1] \dot{\cdot}. Hu[N - 1]].$$

Putting the equation $y = Hx$ together with the decomposition $H = S\Lambda SH$, we have:



We already know one way of understanding how LTI systems operate on signal inputs: they convolve them with the system's impulse response (represented in linear algebra form by the equation $y = Hx$, where H is circulant). The eigendecomposition gives us another understanding. The matrix SH takes the input and extracts what would be the coefficients of the input's representation as a linear combination of harmonic sinusoids (it turns out this is called the discrete Fourier transform). Then, multiplication by the diagonal matrix Λ modifies these coefficients in a way that is particular to the system H (all LTI systems have the same S and SH). Finally, multiplication with the matrix S takes the modified coefficients and expresses them as a linear combination of harmonic sinusoids to give the output y (it turns out that operation is called the inverse discrete Fourier transform).

The Significance of Complex Harmonic Sinusoids

Recall that complex harmonic sinusoids have the form:

$sk[n]=ej2\pi NknN.$

In the context of finite-length discrete-time signals and systems, complex harmonic sinusoids are special, for several reasons. First, the collection of them $\{sk\}k=0N-1$ form a basis for CN, meaning that any signal in CN can be represented as a linear combination of that set of N vectors. Not only this, but they are an orthonormal basis, which means that the weights of the linear combination can be found simply through the inner product of the signal with the harmonic sinusoid in question. And finally, these signals $\{sk\}k=0N-1$ are important because they are eigenvectors of finite-length discrete-time LTI systems. This means that whenever a complex harmonic sinusoid is given as an input to any LTI system, the output is simply a scaled version of that input.

All of these special features come together as we now consider a very important component of the

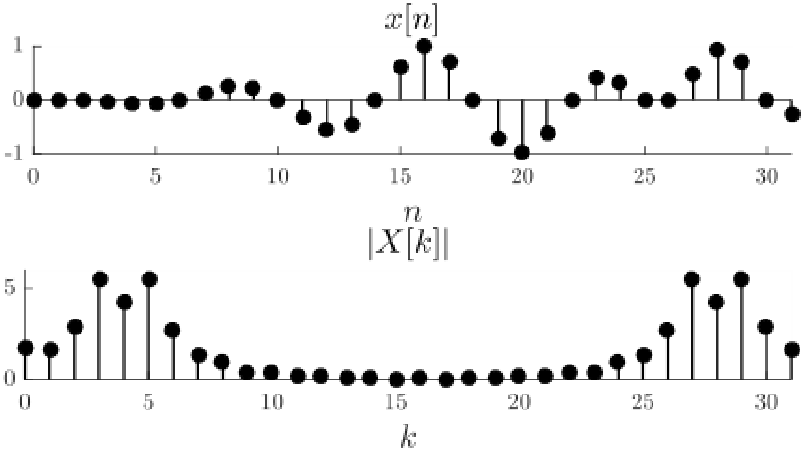
study of signals and systems.
A discrete-time finite length signal $x[n]$, and the magnitude of its DFT, $|X[k]|$.

The Discrete Fourier Transform

In the early 19th century, Jean Baptiste Joseph Fourier showed that any function (later mathematicians would more rigorously qualify that statement) could be composed as a (possibly infinite) sum of harmonic sinusoids. This work resulted in what would be an entire branch of mathematics, Fourier analysis. Fourier analysis extends to many different kinds of signals, including discrete-time finite-length signals. For those, the analysis produces the **discrete Fourier transform (DFT)**. For signals $x[n] \in CN$, the normalized DFT and inverse DFT are:

$$X[k] = \Sigma n = 0N - 1x[n]e - j2\pi NknNx[n] = \Sigma k = 0N - 1X[k]ej2\pi NknN.$$

We see that taking the inner product of $x[n]$ with harmonic sinusoids of different frequencies k (which is what the first sum represents) produces a series of frequency coefficients $X[k]$. That is the DFT. We can also say that it is the ***analysis*** aspect of the Fourier transform, for it gives us a frequency analysis/breakdown of the signal $x[n]$. The frequency coefficients can be used to reconstruct $x[n]$ using the second sum, which is the inverse DFT. It is known as ***synthesis***, for it shows how x[n] can be built up as a linear combination of harmonic complex sinusoids, with the coefficients X[k] telling us how much of each one is needed.



We can express the above signal analysis and synthesis in matrix notation. First we stack up the complex harmonic sinusoids $\{sk\}k=0N-1$ as columns in a matrix S:

$S=[s0|s1|\cdots|sN-1]$.

With these vectors in a matrix, then the signal x is composed of the linear combination of the sinusoids (the synthesis operation), with the weights in a vector X, through the matrix multiplication: $x=SX$ (synthesis; inverse DFT). Finding the weights X, given x, requires the inverse of the basis matrix S. However, since S is unitary, the inverse is simply the Hermitian transpose: $X=SHx$ (analysis; DFT)

Normalized and Un-normalized DFT

To this point, we have been working with the normalized expression of the DFT and its inverse:

- DFT: $X[k] = \Sigma n = 0N - 1x[n]e - j2\pi NknN$
- Inverse DFT: $x[n] = \Sigma k = 0N - 1X[k]ej2\pi NknN$

This form of the DFT has a certain symmetric elegance to it, the only difference in the DFT and its inverse being the negation of the exponent in the complex harmonic sinusoid.

However, it is far more common to use an un-normalized definition of the DFT, one which puts a 1N scaling factor on the inverse, rather than the normalized version splitting this with a 1N in each formula. In practice, the DFT is virtually always understood in terms of the following un-normalized definition:

- DFT: $X[k] = \Sigma n = 0N - 1x[n]e - j2\pi Nkn$
- Inverse DFT: $x[n] = 1N\Sigma k = 0N - 1X[k]ej2\pi Nkn$

Interpretations

So what exactly does the DFT *mean*? There are a variety of interpretations for what the DFT does. We have already explained it in terms of **analysis** and

synthesis. With the inverse DFT, we can build up any signal x[n] as a weighted sum (also known as a linear combination) of complex harmonic sinusoids; this is known as synthesis, which is in accordance with the Merriam-Webster dictionary definition of synthesis as being "the composition or combination of parts or elements so as to form a whole." But how are we to know how *much* of each sinusoid goes in to the synthesis? The DFT tells us the coefficient values, one for each of the N sinousids. This is the analysis aspect of the DFT, for it "analyzes" the signal in terms of the frequencies in it, how much there is of every possible frequency within the signal. To use a cooking metaphor, the analysis tells us the amount of each ingredient that goes into the dish that is our signal, and the cooking process is the synthesis which takes the ingredients and creates the finished product.

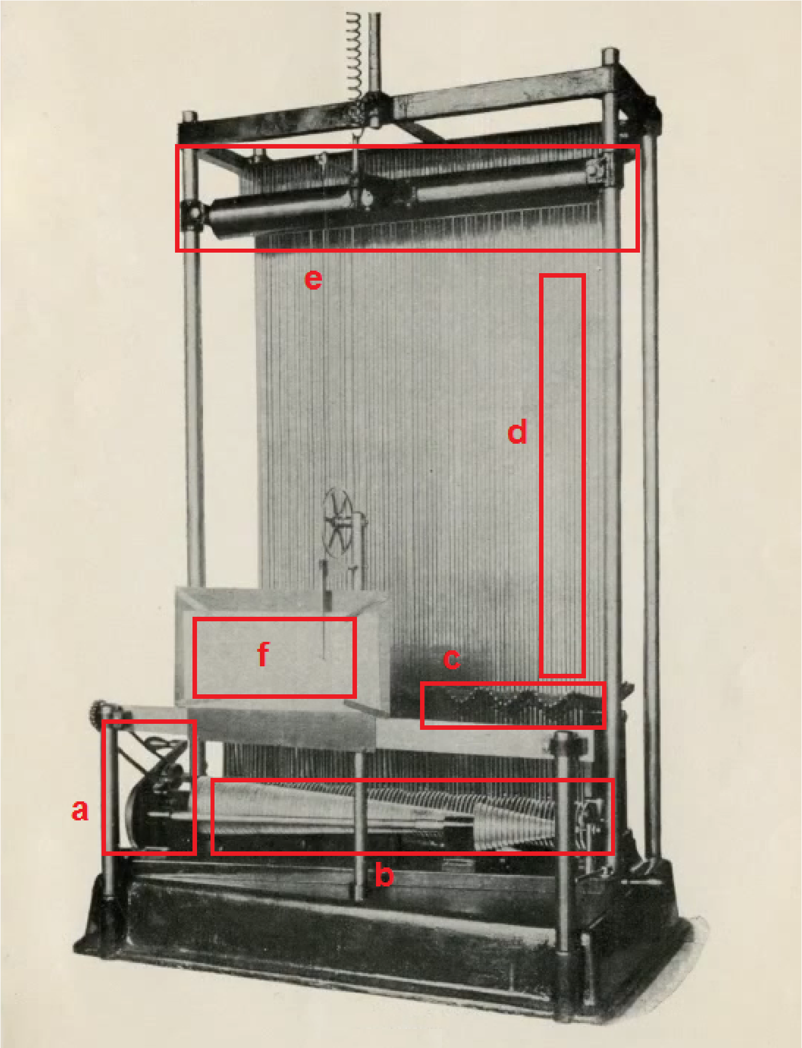
There is another way of looking at the DFT and inverse DFT. We note that there is a one-to-one correspondence between any signal x[n] and its DFT X[k]; a signal x[n] has only one DFT, and any X[k] has only one inverse. So x[n] and X[k] are really referring to the same one thing--some signal--but only in different ways. x[n] describes the signal in terms of its value at every given time location n, while X[k] describes the signal in terms of how much of each frequency contributes to it. So there is a *time-domain* description of the signal, which is x[n], and a *frequency-domain* description of it as

well, which is X[k]. We could think of the time-domain and frequency-domain as two different languages, like English and Spanish. For some kind of idea or expression, it can be represented in either one language or another, and given the expression in one language, it can be translated to the other. Of course, this is not a perfect metaphor; spoken languages are not always one-to-one (and some words in one language do not have an adequate word in another language). What this interpretation does convey is that time and frequency are two different, and yet also equivalent, ways of expressing a signal entity.
Heideman, Michael, Don Johnson, and C. Burrus. "Gauss and the history of the fast Fourier transform." IEEE ASSP Magazine 1, no. 4 (1984): 14-21. The a) crank, b) gears, c) arms, d) levers, and e) springs of a harmonic analyzer can together either synthesize a function or analyze it, writing the resultant plot on a f) piece of paper.

Excursus: A Physical Implementation of Fourier Analysis and Synthesis

The inverse DFT shows us that a time-domain signal can be represented as a summation of complex sinusoidal functions, while the DFT reveals how much of each complex sinusoid is required in the summation. Before the advent of computers, calculating either the DFT or inverse DFT was obviously a tedious process. Even with the use of some algorithmic shortcuts (e.g., by Carl Friedrich Gauss in the early nineteenth century[footnote]), the calculations had to be done by hand.

But "by hand" doesn't have to mean writing out the equations and consulting mathematical tables. In the early twentieth century, Albert Michelson invented a machine that could perform Fourier synthesis and analysis. This "harmonic analyzer" used a series of gears, levers, and springs to physically implement the inner products we see in the DFT and inverse DFT equations.



Recall that in synthesis, the inverse DFT adds together a weighted sum of complex sinusoidal functions. The machine performs a similar operation, with each part of the machine corresponding to an aspect of the inverse DFT formula. The crank of the machine rotates a series of gears that each turn another gear. Each of these gears would correspond with a certain k of the formula, with the turning of the crank being the equivalent of progressing through the sinusoids in time (the variable n). The rightmost gear would have a single revolution per turn of the crank, then the next one over, two revolutions, and so on. All of these gears are connected with rods to a set of arms. As the gears turn, the arms rock up and down. Thus turning the crank produces a set of sinusoids of varying frequency (in the figure above, from $k=1$ to $k=80$!).

Of course, the synthesis operation must weight these sinusoids. The sinusoids are weighted (the $X[k]$ values in the inverse DFT formula) by setting each lever at a specified position along the rocker arm. The farther towards either the edge of the arm, the more the lever will move along with the arm. At the top of the machine, the ends of these levers are attached to springs that are then attached to a bar with a single counter-spring. All of these springs being connected to the same pivoting bar "sums" the movement of all of the levers, which corresponds to the sum Σ in the formula. At the end of the pivoting bar is a lever that is tied to a pen, which moves up and down as a piece of paper tranverses left to right while the crank turns. Thus the pen produces on the paper the output of the inverse DFT summation.

The machine can also be used to perform analysis. In order to find the proper weights that would produce a desired function through analysis, a single period of the function would be "inputted" into the machine by setting the lever positions across all of the arms into the shape of the function. That shape across all the arms corresponds to the x[n] of the DFT formula, and now the number of arms correspond to the time variable n. As the crank is turned, the arms will make a shape of a sinusoid (as you can see in section c) of the figure). One turn of the crank will produce a very long wave, a single sinusoidal period across all the bars, i.e. a sinusoid with frequency $k=1$. The shape of the function weighted across all of these bars corresponds to the weighting in the DFT formula, and once again the pivoting bar will sum everything together. After one turn, the position of the pen will correspond to X[1], after two turns, X[2], and so on, thereby computing the DFT.

To see a video of the harmonic analyzer at work, visit http://www.engineerguy.com/fourier/.

Discrete Fourier Transform Properties

Recall, for discrete-time finite length signals, the definition of the DFT and the inverse DFT (given here in its normalized form):

- DFT: $X[k] = \Sigma n = 0 N - 1 x[n] e - j 2 \pi N k n N$
- Inverse DFT: $x[n] = \Sigma k = 0 N - 1 X[k] e j 2 \pi N k n N$

Recall also its much more commonly used un-normalized form:

- DFT: $X[k] = \Sigma n = 0 N - 1 x[n] e - j 2 \pi N k n$
- Inverse DFT: $x[n] = 1 N \Sigma k = 0 N - 1 X[k] e j 2 \pi N k n$

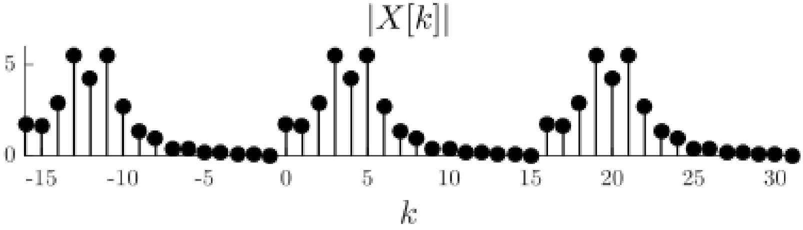
A signal $x[n]$ and its DFT $X[k]$ (recall there is a one-to-one correspondence) are referred to as a **DFT pair**. The DFT has a variety of properties, which we will now consider.
 The DFT $X[k]$ of any N-length signal is periodic, with period N.

The DFT and Its Inverse Are Periodic

The DFT is defined for finite-length (length N) signals; so, for $x[n]$, n runs from 0 to $N-1$, as does k. But let's see what happens when we consider a value of k outside this range in the DFT formula, say $X[k+lN]$, where l is some nonzero integer:

X[k+lN]= Σn=0N−1x[n]e−j2πN(k+lN)n=Σn=0N−1x[n]e−j2πNkne−j2πNlNn=Σn=0N−1x[n]e−j2πNkn·1=Σn=0N−1x[n]e−j2πNkn=X[k].

As X[k+lN]=X[k], the DFT is periodic:



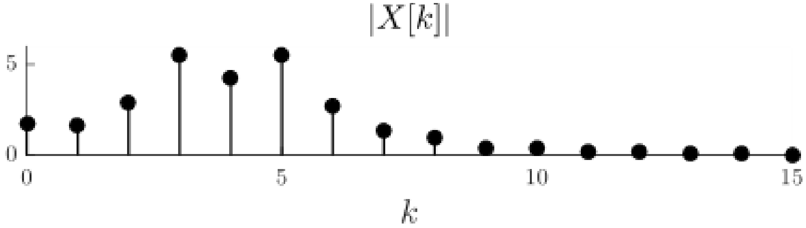
By the same token, the inverse DFT is also periodic:

x[n+lN]= 1NΣk=0N−1X[k]ej2πNk(n+lN)=Σk=0N−1X[k]ej2πNknej2πNklN=Σk=0N−1X[k]ej2πNkn·1=Σk=0N−1X[k]ej2πNkn=x[n].

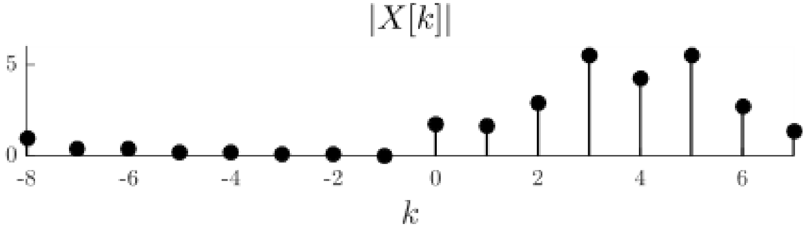
Again this is to be expected because the complex harmonic sinusoids of the inverse DFT sum are periodic. This also further illustrates the fact that any finite-length signal can be understood as a single period of a corresponding periodic signal.
For an N=16 length signal, one way to express the range of frequencies is for k to run from 0 to N−1, which corresponds to frequencies between 0 and 2π.
For an N=16 length signal, another way to express the range of frequencies is for k to run from −N2 to N2−1, which corresponds to frequencies between −π and π.

DFT Frequency Ranges

A complex harmonic sinuoid ej(2πNk)n has, by definition, a frequency of 2πNk, which may label as ωk. In the DFT of a signal of length N, the variable k ranges from 0 to N−1, values that correspond to frequencies from 0 to (just about) 2π:



However, as we saw above, since the DFT is periodic, 0 to N−1 is not the only range of k we may use to express the DFT. Since X[k] = X[k−N], we may let k run from −N2 to N2−1 (for even N, that is; for odd N it would be −N−12 to N−12):



Shifts in Time and Frequency

Let x[n] and X[k] be a DFT pair (i.e., X[k] is the DFT of x[n]). A circular shift in time on x[n] will produce a phase shift on X[k]:

x[(n − m)N] ⟷DFT e − j2πNkmX[k].

To prove this relationship, we first note that for the circular shift (n − m)N, there is some integer l such that(n − m)N = n − m + lN. We will use that fact for a change of variables in our proof:

DFT{x[(n − m)N]} = Σn = 0N − 1x[(n − m)N]e − j2πNknLet r = (n − m)N = n − m + lN = Σr = 0N − 1x[r]e − j2πNk(r + m − lN) = Σr = 0N − 1x[r]e − j2πNkre − j2πNme − j2πNlN = e − j2πNmΣr = 0N − 1x[r]e − j2πNkr·1 = e − j2πNkmX[k].

As you might expect from the symmetrical similarities between the DFT and inverse DFT, a comparable relationship exists in the other direction as well. A circular shift in the frequency domain of a signal results in the modulation of the signal in the time domain:

ej2πNlnx[n] ⟷DFT X[(k − l)N]. The proof of this property follows the same approach as that of the first.

The DFT is Linear

As the DFT of a signal is essentially a weighted sum, it follows naturally that the DFT operation is linear. So if we have two arbitrary signals x1[n] and x2[n] (with DFTs X1[k] and X2[k]) and arbitrary constants $\alpha1$ and $\alpha2$, then the DFT of $\alpha1x1[n]+\alpha2x2[n]$ is simply $\alpha1X1[k]+\alpha2X2[k]$:

$$x1[n] \longleftrightarrow \mathrm{DFT}\ X1[k],\ \ x2[n] \longleftrightarrow \mathrm{DFT}\ X2[k]\alpha1x1[n]+\alpha2x[2]$$

The proof is straightforward:

$$\begin{aligned}
&\mathrm{DFT}\{\alpha 1 x 1 [n] + \alpha 2 x 2 [n]\} = \Sigma n = 0 N \\
&- 1 (\alpha 1 x 1 [n] + \alpha 2 x [2]) e - j 2 \pi N k n \\
&= (\alpha 1 \Sigma n = 0 N - 1 x 1 [n] e - j 2 \pi N k n) \\
&+ (\alpha 2 \Sigma n = 0 N - 1 x 2 [n] e - j 2 \pi N k n) \\
&= \alpha 1 X 1 [k] + \alpha 2 X 2 [k]
\end{aligned}$$

The Convolution/Multiplication Time/Frequency Relationship

We now reach perhaps the most significant DFT property, the relationship between convolution and multiplication in time and frequency. Suppose we have two discrete-time finite length signals $x1[n]$ and $x2[n]$, whose DFTs are $X1[k]$ and $X2[k]$. Let $y[n]$ be the circular convolution $y[n]=x1[n]\circledast x2[n]$. Then the DFT of $y[n]$ is equivalent to the product of the DFTs of $x1[n]$ and $x2[n]$: $Y[k]=X1[k]X2[k]$. Or, to express this relationship in DFT pair notation, we have:

$$x 1 [n] \circledast x 2 [n] \longleftrightarrow \mathrm{DFT}\ X 1 [k] X 2 [k]$$

The proof of this relationship will use the $r=(n$

−m)N=n−m+lN change of variable we used earlier:

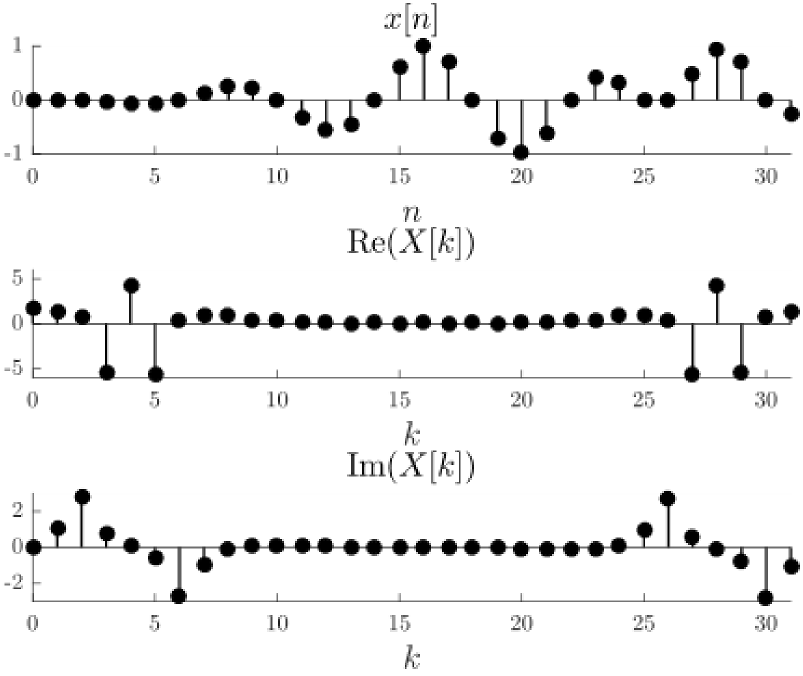
DFT{x1[n]⊛x2[n]}= Σn=0N−1(Σm=0N−1x1[(n−m)N]x2[m])e−j2πNkn=Σm=0N−1x2[m](Σn=0N−1x1[(n−m)N]e−j2πNkn)=Σm=0N−1x2[m](Σr=0N−1x1[r]e−j2πNk(r+m+lN))=Σm=0N−1x2[m](Σr=0N−1x1[r]e−j2πNkre−j2πNkme−j2πNklN)=(Σm=0N−1x2[m]e−j2πNkm)(Σr=0N−1x1[r]e−j2πNkr)=X1[k]X2[k].

So we see that convolution in two signals' time corresponds to their multiplication in frequency. It is certainly an interesting relationship, but it is also very practical: N multiplications in the frequency domain are obviously fewer than the N2 multiplications required to compute for an N-length signal. Of course, this requires first that the DFT of the two signals be computed (and then the inverse DFT of the product), but we will see there are ways to compute that efficiently. What it all means is that a system's output (which is found via convolution) can be computed efficiently by way of multiplication in the frequency domain.

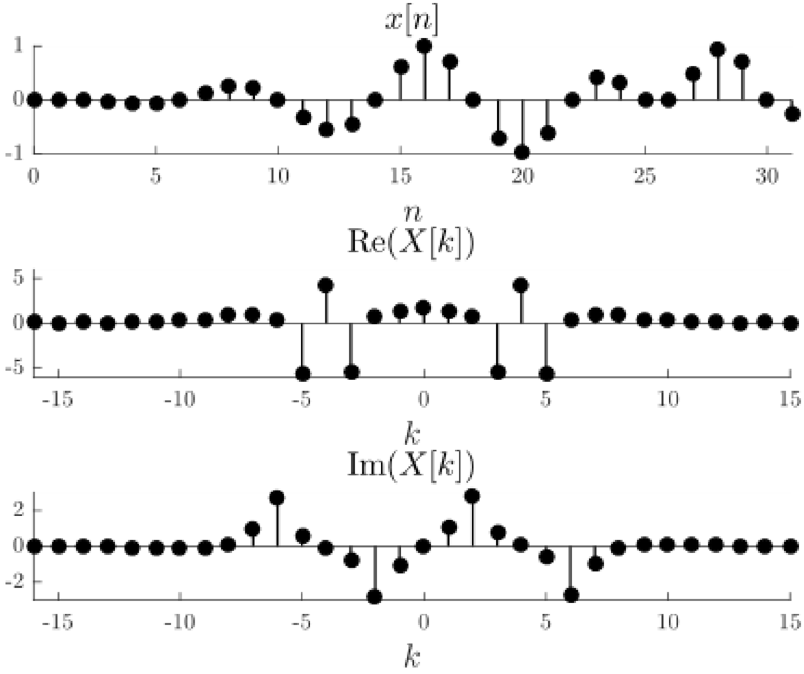
A real-valued finite-length signal x[n], with the real and imaginary parts of its DFT.For real-valued signals such as x[n] here, the real part of their DFTs are even and the odd part is odd, which is more readily seen when they are plotted with k ranging from −N2 to N2.

DFT Symmetry Properties

As the DFT is a weighted sum of complex signals (complex harmonic sinusoids), it follows that the DFT of signals is, in general, complex-valued.



Perhaps you have noticed, in the figures above, that there is a certain symmetry for these (real-valued) signals' DFTs. For the real part of the DFT, you see that $\mathrm{Re}\ [X[k]] = \mathrm{Re}\ [X[N-k]]$, and that $\mathrm{Im}\ [X[k]] = -\mathrm{Im}\ [X[N-k]]$. Recalling that the DFT is periodic, if we consider frequencies from $-N2$ to $N2$, then we have $\mathrm{Re}\ [X[k]] = \mathrm{Re}\ [X[-k]]$, and that $\mathrm{Im}\ [X[k]] = -\mathrm{Im}\ [X[-k]]$. Or, in words: the real part of the DFT is even, and the imaginary part is odd.



This symmetry was not a coincidence for the DFT of that particular x[n]; because the complex harmonic sinusoid signals that make up the DFT sum formula are conjugate symmetric (the real part is even, the imaginary part is odd), the DFTs of all purely real signals will also be conjugate symmetric. The converse is true for purely imaginary signals; their DFTs will be symmetric such that the real part is odd and the imaginary part is even. If, in addition to being purely real or purely imaginary, the signal itself is also even or odd, that will further limit the characteristics of its DFT. Note that if a signal is complex-valued, its DFT will not--in general--exhibit any such symmetries.

$x[n]$	$X[k]$	$\mathrm{Re}(X[k])$	$\mathrm{Im}(X[k])$	$\lvert X[k]\rvert$	$\angle X[k]$
real	conjugate symmetric: $X[-k] = X[k]^*$	even	odd	even	odd
real and even	real and even	even	zero	even	$0, \pi$
real and odd	imaginary and odd	zero	odd	even	$\pm \pi 2$
imaginary	conjugate odd symmetric $X[-k] = -X[k]^*$	odd	even	even	odd
imaginary and even	imaginary and even	zero	even	even	$\pm\pi 2$
imaginary and odd	real and odd	odd	zero	even	$0, \pi$

Tasks and Algorithms

The first thing to know about the **fast Fourier transform**, or **FFT**, is that it actually is not a unique type of mathematical transform at all. The FFT is simply a method, or *algorithm*, for the task of computing a DFT. In that way, it is similar to the task of sorting a group of numbers from smallest to largest. No matter how you might go about the task of sorting the numbers, there is only one correctly sorted answer. Some methods to sorting, however, may be quicker (take fewer steps) than others. For example, you could put the numbers in one list, and then progressively traverse through the list, comparing two adjacent items at a time and swapping them if necessary; each time you traverse the list, you will have put the next largest number in its place. This approach is usually called a "Bubble Sort." Another approach is to split the list of numbers into two sets, then divide each set in two, and divide those sets, and so on, until each little set has two (or fewer) numbers. You will then sort each set of two numbers (easy!), and then progressively combine these sorted sets together (also easy!) until a single sorted set remains. This is called a "Merge Sort," and is known as a "divide and conquer" algorithm.

Now, these two sorting methods achieve the same results, but the Merge Sort typically does so in far fewer steps (although the Bubble Sort will process an already or nearly sorted list quicker). If there are N items in a list, Merge Sort requires about Nlog2N steps, while the Bubble Sort requires about N2. Even with lists as small as 32 items, that makes a big difference, the difference between 160 and 1024. So it is with the discrete Fourier transform. The DFT is a simple formula:

$$X[k]=\Sigma n=0N-1x[n]e-j2\pi Nkn.$$

It would be possible to calculate this as a sum of N multiplications (going from $n=0$ to $n=N-1$), doing that N times (for each k, again from 0 to $N-1$). That's about N2 multiplications and additions. But, there are many different ways one could calculate the DFT, just as there are many ways to sort a list. As with Merge Sort, there is a divide-and-conquer approach to finding the DFT which does so with far fewer computational steps, only about Nlog2N.

The FFT is a "divide and conquer" algorithmic approach to finding a DFT. An N-length DFT can be found as the weighted sum of two N/2-length DFTs, and each of those are the sum of two N/4-length DFTs, and so on.The DFT of a length-8 signal can be split into the the weighted sum of two length-4 DFTs, namely, the DFTs of the even and odd indices. Each length-4 DFT is split into the weighted sum of two length-2 DFTs. The length-2 DFTs are found, and from there the rest of the overall DFT can be computed according to the half-size weighted sums. The 2-length DFTs are simple to find, just an addition and a subtraction. When those operations are represented graphically, they are said to have a "butterfly" structure.

The FFT: A Divide and Conquer Algorithm for Finding a DFT

The intuition behind the FFT algorithm is the same as that of the Merge Sort. The Merge Sort operates on the idea that it is easy to take two sorted lists and combine them into one large sorted list: you simply compare the beginning of each list, pick off the smallest of the two and place it in a new list, and then keep on doing that until the two lists are empty.

Something similar happens with the DFT. Suppose you have some signal $x[n]$, which you can split into two smaller signals in terms of its even and odd indices, $x[2n]$ and $x[2n+1]$. Suppose now that you already know the DFTs of these two smaller signals, and call them $E[k]$ and $O[k]$. Then it is easy to find the DFT of the whole signal, $X[k]$, from these smaller ones:

$X[k] = E[k] + ej2\pi NkO[k]$.

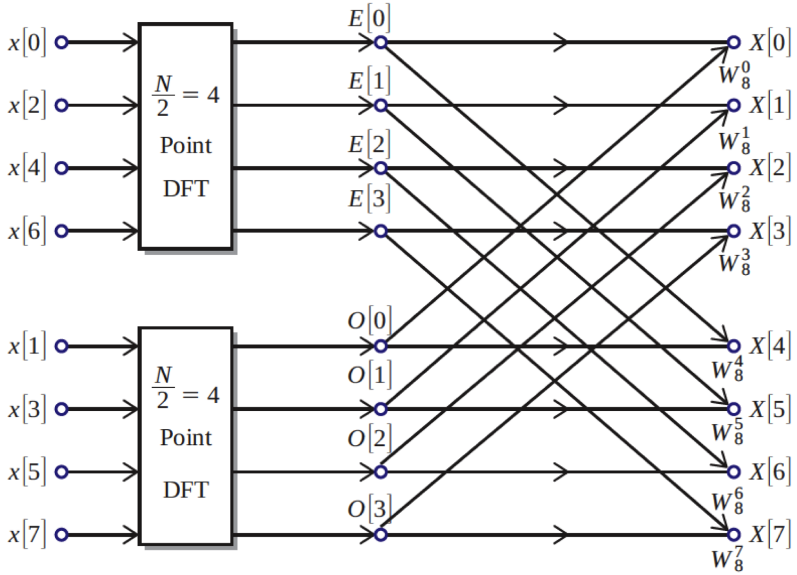
A proof of this follows. For simplification of notation, we let WN = e − j2πN:

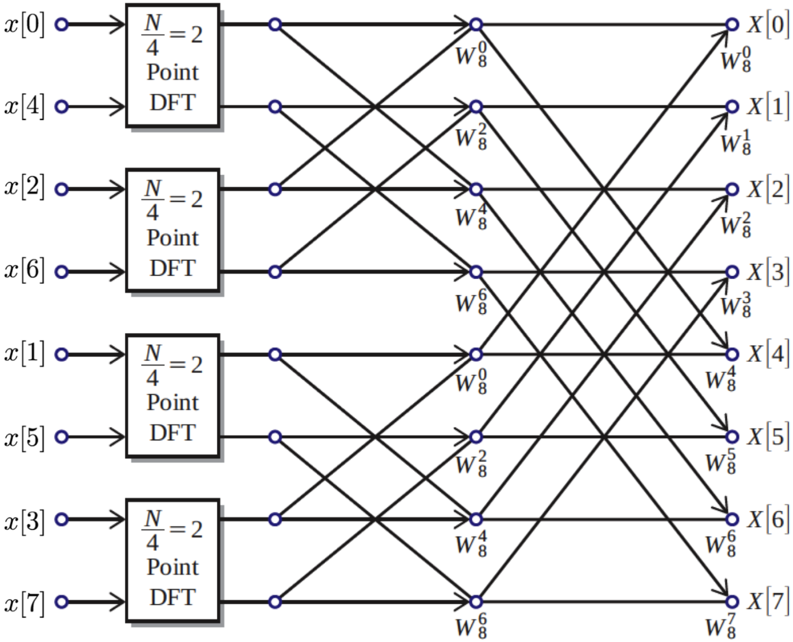
X[k] = Σn = 0N
− 1x[n]WNkn = Σn = 0N/2 − 1x[2n]WNk(2n) + Σn = 0N/2
+ 1]WNk(2n
+ 1) = Σn = 0N/2 − 1x[2n]WN2kn + WNkΣn = 0N/2 − 1x[2
+ 1]WN2kn = Σn = 0N/2 − 1x[2n]WN/2kn + WNkΣn = 0N/
+ 1]WN/2kn = E[k] + WNkO[k].

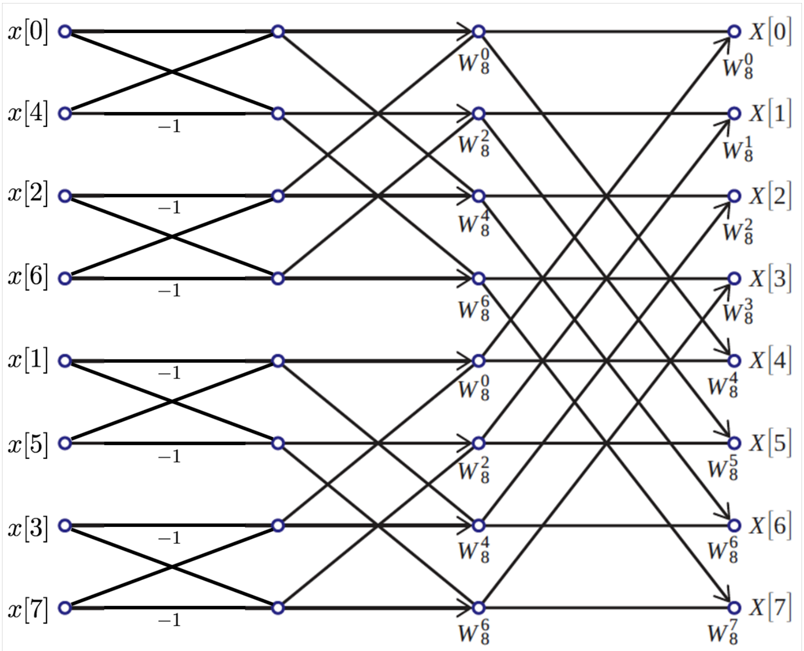
You will note that the k in X[k] runs from 0 to N − 1, but that the even and odd DFTs are only of length N/2. It may seem a little strange to consider the values of these DFTs for k greater than N/2 (as is necessary in the formula). But there is no problem with this, since DFTs are periodic; as those DFTs are length N/2, we have E[N/2 + k] = E[k] for those values between N/2 and N − 1. So, we have that the DFT of an N-length signal is simply the weighted sum of two N/2-length DFTs. If we, somehow, are given the DFT of those even/odd sub-signals, there are only N additions and multiplications involved in finding the DFT of the whole signal, which is of course much better than the N2 needed to find the DFT by straightforward calculation of the DFT sum.

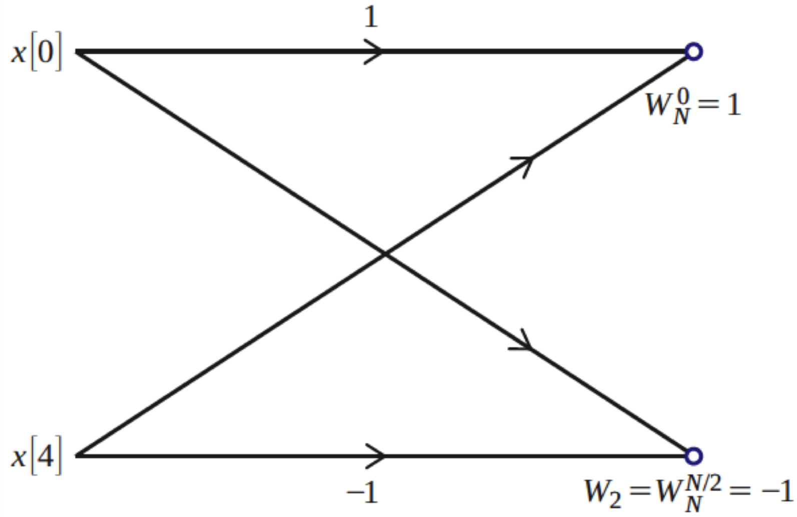
Of course, this requires that we had the DFTs of the sub-divided signals! How do we find those? Well, again we will divide those sub-signals by even/odd indices, and then do that again, and again, and again, until we have lots of length-2 sub-signals. The DFT of a length-2 signal is *very* easy to compute. If we call such a signal $t[n]$, then $T[0] = t[0] + t[1]$ and $T[1] = t[0] - t[1]$ (this is easy to verify for yourself).

With the DFTs of our many length-2 signals in hand, we use the even/odd sum formula to combine them find the DFT of the length-4 signals, and then length-8, and so on until we have the DFT of the whole signal. This is known as a decimation-in-time FFT, and while it assumes a signal length that is a power of 2, there are ways to apply the approach to signals of other lengths. The figure below illustrates the process of splitting up the DFT into smaller and smaller half-size DFTs:




x[0]
x[4]
x[2]
x[6]
x[1]
x[5]
x[3]
x[7]
N/4 = 2 Point DFT
N/4 = 2 Point DFT
N/4 = 2 Point DFT
N/4 = 2 Point DFT
W_8^0
W_8^2
W_8^4
W_8^6
W_8^0
W_8^2
W_8^4
W_8^6
X[0]
X[1]
X[2]
X[3]
X[4]
X[5]
X[6]
X[7]
W_8^0
W_8^1
W_8^2
W_8^3
W_8^4
W_8^5
W_8^6
W_8^7


x[0]
x[4]
x[2]
x[6]
x[1]
x[5]
x[3]
x[7]
−1
−1
−1
−1
−1
−1
−1
W_8^0
W_8^2
W_8^4
W_8^6
W_8^0
W_8^2
W_8^4
W_8^6
X[0]
X[1]
X[2]
X[3]
X[4]
X[5]
X[6]
X[7]
W_8^0
W_8^1
W_8^2
W_8^3
W_8^4
W_8^5
W_8^6
W_8^7



Computational Savings of the FFT

Recall that a straightforward sum and multiplication computation of the DFT requires about N2 operations; to be precise, it is N2 multiplications and N2 − N additions. The FFT approach requires about Nlog2N operations: Nlog2N − N multiplications and Nlog2N additions. For small signal sizes, the difference between N2 and Nlog2N is not that significant. But as N gets even moderately large (say, N = 32 and up), the difference becomes larger and larger. By the time we consider typical signal sizes in signal processing, in which N is in the millions, taking N2 complex operations is prohibitive in terms of time and memory costs, while Nlog2N is very manageable.

Fast Convolution with the FFT

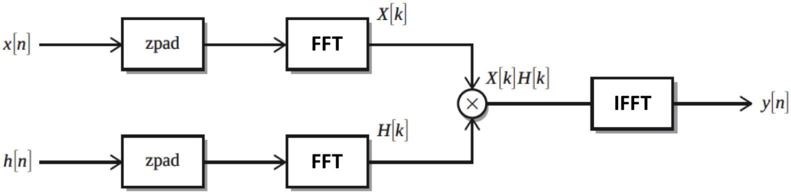
We have seen, previously, these four important signal processing facts:

- the output of infinite-length LTI systems can be found via linear convolution
- linear convolution can be found through circular convolution, through zero-padding
- there is an equivalence between circular convolution in the signals' time domain and multiplication in their frequency domain
- the FFT is an algorithm that can quickly compute a DFT

Perhaps at the time it may have seemed that these findings were unrelated to each other. But as we now string them together, you can see the incredibly significant consequence: the output of LTI systems can be computed very efficiently. This truth has supported the incredible advances in signal processing over the past fifty years.
 Through zero-padding and forward and inverse DFTs, using the FFT algorithm, an LTI system's output can be found efficiently.

Putting It All Together...

Suppose we have an LTI system with an impulse response of h[n] and an input of x[n]. We would like to find the output, y[n]. We could find this output via linear (infinite length) convolution. If the length of the impulse response is Nh and that of the input signal is Nx, then about NhNx operations would be required to compute this convolution. It would also be possible to compute the output through circular convolution, by zero-padding each signal to be of length $Nh + Nx - 1$, and then circularly convolving the zero-padded signals. This would not save any computational steps, but it is a significant insight because the circular convolution could be performed by using DFTs: simply take the DFT of each zero-padded signal, multiply the two DFTs together, then take the inverse DFT of the result. Again, this does not at first seem to save any computational steps, and in fact seems to add even more, except that we have seen that the FFT is able to perform DFTs in about Nlog2N operations. What that means is that if we zero-pad x[n] and h[n], then take the DFT of each (using the FFT algorithm), multiply these two together, then take the inverse DFT of the result, we can find the system output in about $2(Nh + Nx - 1)\mathrm{log2}(Nh + Nx - 1)$ operations. As signal lengths increase, this can end up being huge computational savings over the NhNx operations it would take to find the output through a convolution sum. Below is a graphical depiction of the steps to quickly find the output, with the help of the FFT:


$x[n]$
zpad
FFT
$X[k]$
$h[n]$
zpad
FFT
$H[k]$
$X[k]H[k]$
IFFT
$y[n]$

Other Orthogonal Bases

We have seen that the discrete Fourier transform (DFT) is an example of an orthogonal basis for finite length discrete-time signals. As a basis, a linear combination of its basis elements {bk} can represent any signal x:

$$x = \Sigma k = 0N - 1\alpha kbk = Ba.$$

Furthermore, as an orthogonal basis, its basis elements are mutually orthogonal, meaning that the weights needed to build up a particular signal x can be found with a weighted inner product of x and the basis element in question. If all of the basis elements are of unit-norm, then the basis is orthonormal, and a simple inner product will find the weights:

$$\alpha k = \langle x,bk \rangle \text{ , } a = BHx.$$

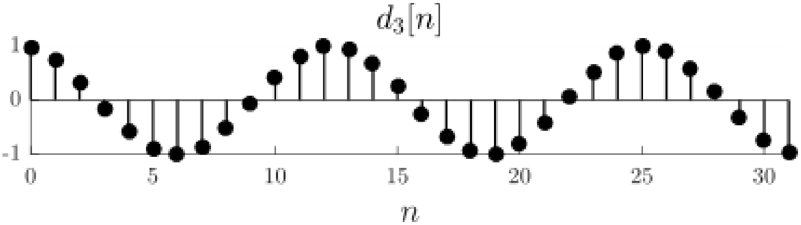
The DFT is a particularly useful basis, for its basis vectors $ej2\pi Nkn$ are the eigenfunctions of all LTI systems. As such, they are simply scaled when passing through a system. But as useful as the DFT is, there are other significant orthogonal bases to consider, as well.

The real part of the DFT functions. The imaginary part of the DFT functions. The DCT functions are real-valued. For the this particular signal x[n], note now the real and imaginary parts of its DFT have significantly more large values than its DCT does.

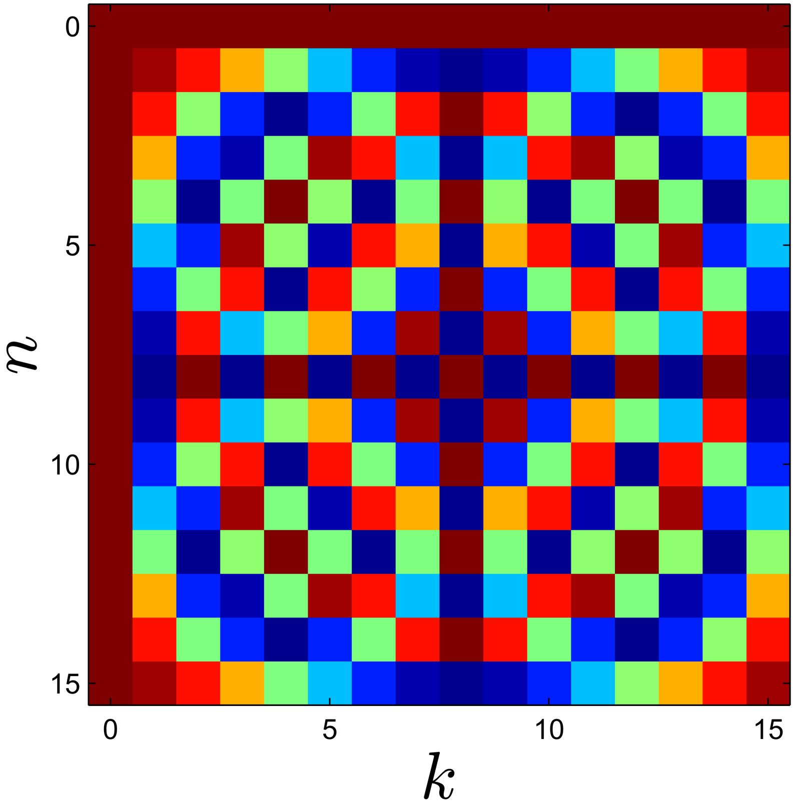
The Discrete Cosine Transform

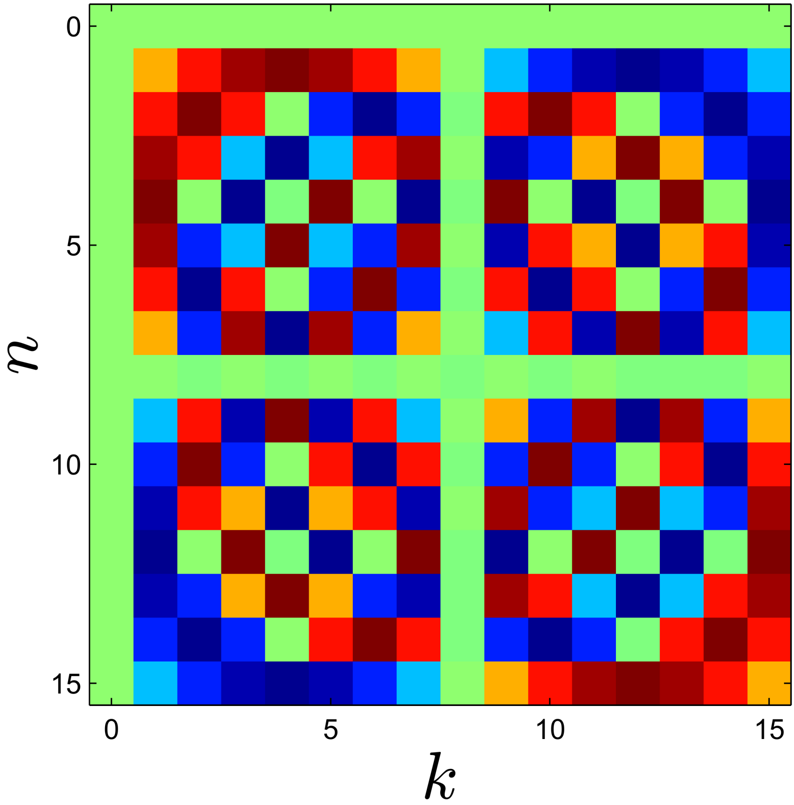
Since the DFT uses complex harmonic sinusoids to analyze and synthesize signals, the DFT coefficients of real signals are almost always complex-valued (unless the signal is even, in which case the coefficients are purely real). This normally poses no problem, but there are some circumstances in which real coefficients are preferable. One example is when signal compression is desired. In such applications, a signal is transformed and only a few of the largest transform coefficients are kept. Having to store both the real and the imaginary parts of the coefficients would mean twice the storage. It is for this reason that the discrete cosine transform (DCT) is used in signal compression applications (such as the JPEG image compression standard). The DCT is very similar to the DFT, as it also uses sinusoids as basis elements, except that its basis elements are purely real-valued. There are several varieties of the DCT, but the most popular is the DCT-II, which has the following basis elements:

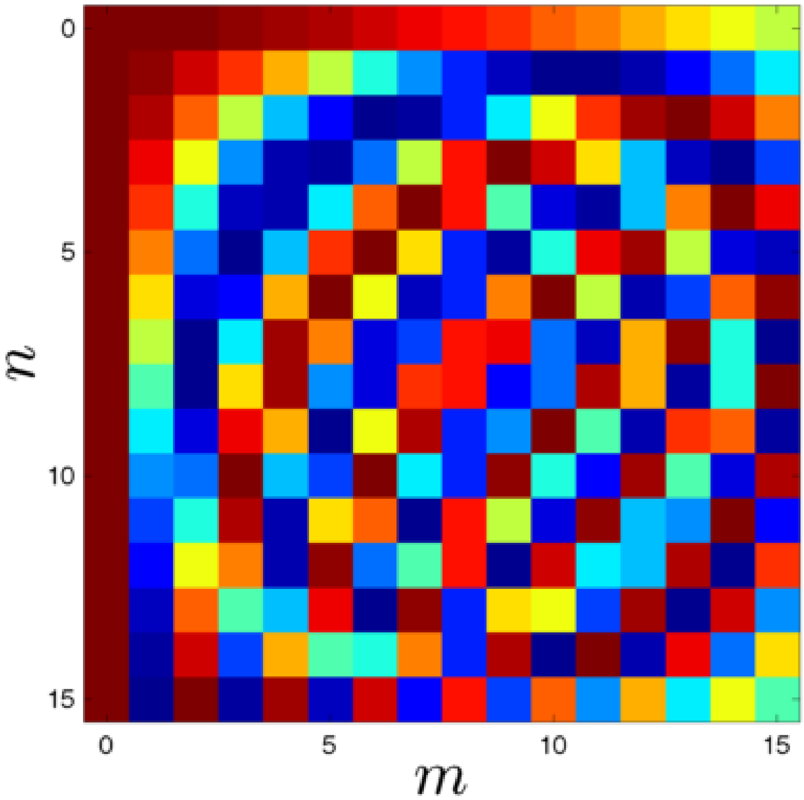
$dk[n] = \cos[\pi N(n+12)k], 0 \leq n,k \leq N-1.$



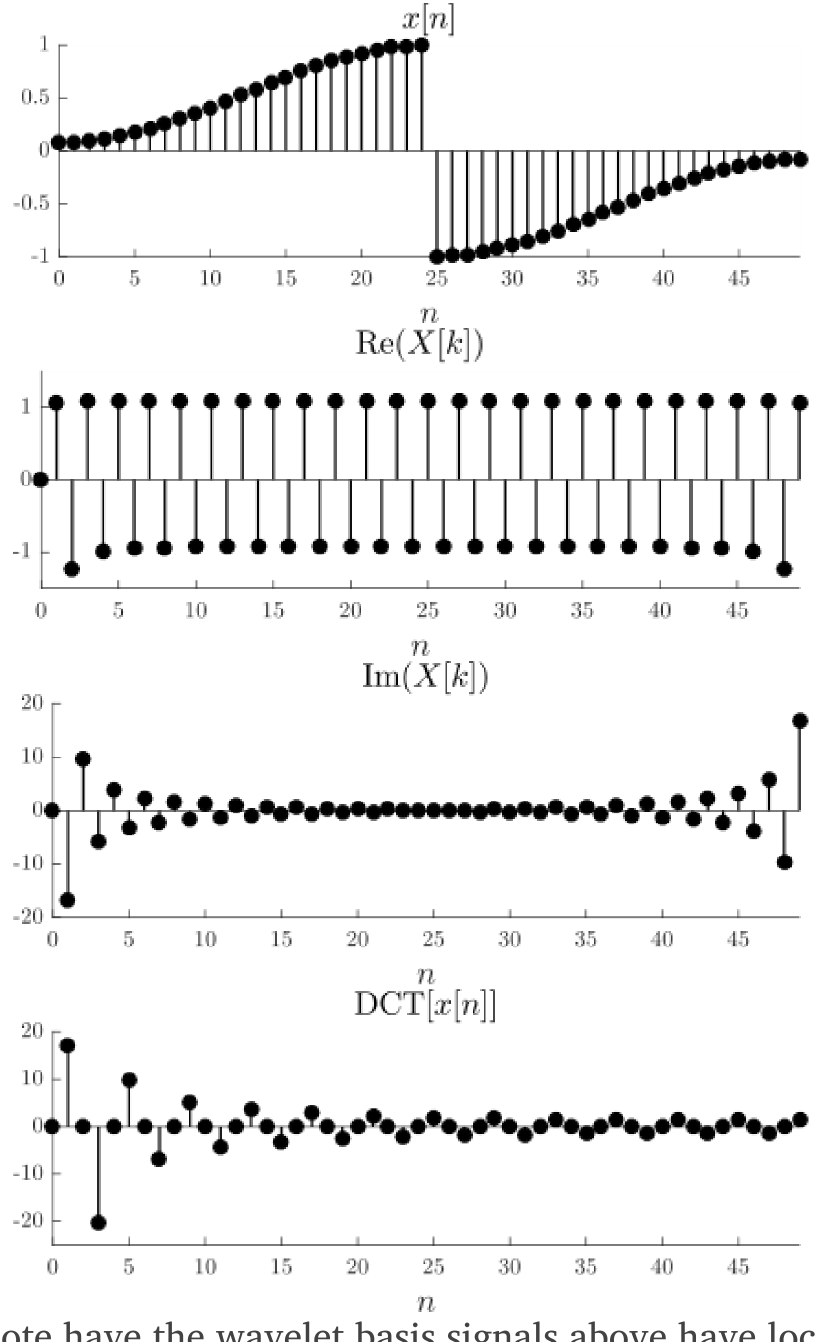
Unlike the DFT basis functions, the DCT functions are completely real-valued.




0
5
10
15
n
0
5
10
15
k



Since the DCT basis functions are real valued, the DCT transform coefficients for a real signal are also real valued. As a result, there are cases in which signals can be represented reasonably well when reconstructed with just a handful of DCT coefficients, compared to the same number of DFT coefficients.



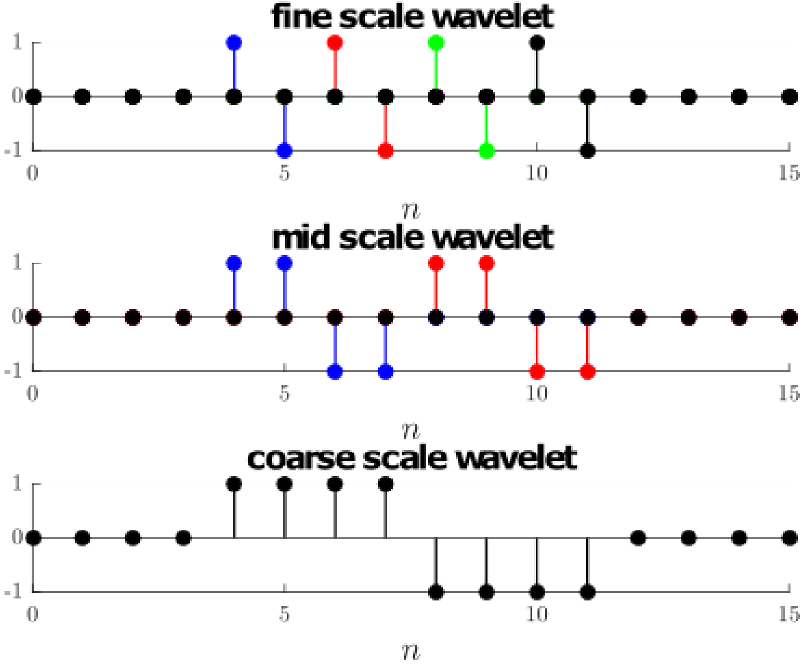
Note have the wavelet basis signals above have local sinusoidal features. They are thus useful for representing signals with similar characteristics.

Wavelet Transforms

We have seen that with the DFT (and DCT as well), signals can be represented in either the time domain (e.g., as $x[n]$), or in the frequency domain through their DFT (e.g., $X[k]$). Both representations are equivalent, of course, for one can always be derived from the other. But sometimes one way of expressing a class of signals is desired, such as when as signal can be expressed with just a few values. For example, the signal $x[n] = \delta[n]$ can be expressed with just a single nonzero value in the time domain, but it does not have any nonzero values in the frequency domain. Thus we say the signal is localized in time. Likewise, the signal $x[n] = ej2\pi 16n$ has only a single nonzero DFT coefficient, while all of its values in the time domain are nonzero. That is a kind of signal we say is localized in frequency. If a signal is localized in time or in frequency, we can efficiently store and process its signal values by storing/operating on the signal in the best of those two domains.

But some signals are not localized very well in time *or* in frequency. Supposing a signal has frequency components that show up and then are absent at different times, then neither the time nor the frequency domain does a good job of efficiently representing such as signal. An example of such a signal would be a musical melody; at any given time it may be expressed as being a single (predominant) frequency, but these frequencies (i.e., the notes) obviously change quite a bit over time (the work of Philip Glass notwithstanding).

There are many transforms that are designed specifically for analysis and concise representation of these signals that have localized frequency components, two of which we will consider. The first is the Haar Wavelet, which is a type of wavelet transform. The basis functions of the Haar wavelet transform, like other wavelets, are localized in both time and frequency, as illustrated in the following figure:



Unlike the DFT or DCT, wavelet basis functions are nonzero for most of their duration in time. But unlike delta functions, they have particular frequencies for their duration. So in a sense they have the benefits of both domains, combined.

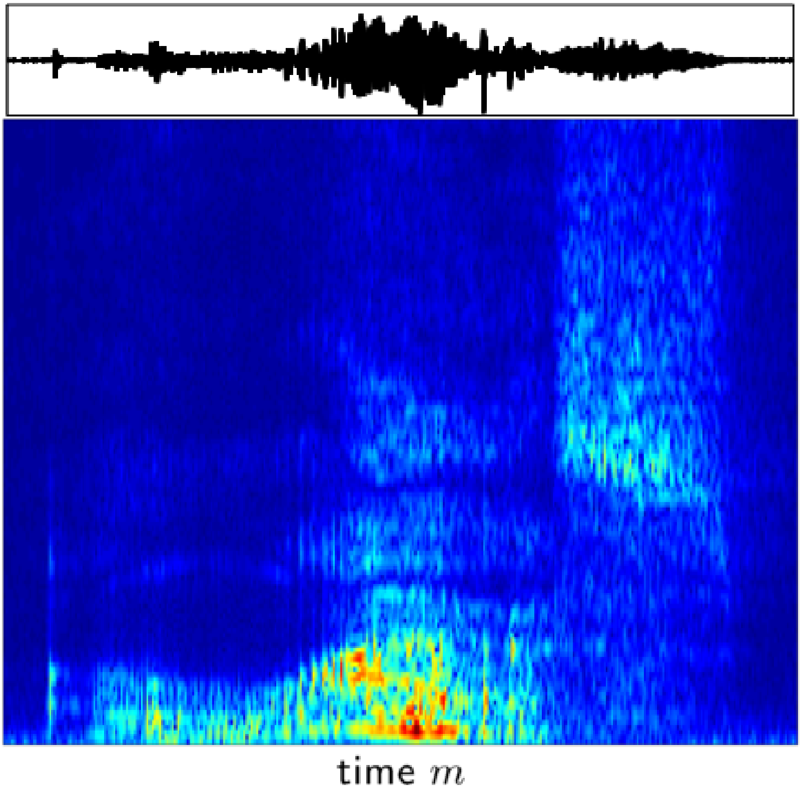
A STFT of a speech sample.

The Short Time Fourier Transform

Another transform that considers signals with time-localized frequency components is the short-time Fourier transform (STFT). The STFT is kind of like a DFT, except that it first splits a long signal into many smaller parts, and then takes a DFT of each of those parts. This "splitting" is done by multiplying the signal (of length N) by a moving window (of length Nw), progressively taking the DFT of the product:

$X[m,k]=\Sigma n=0N-1x[n]w[n-m]ej2\pi Nwk(n-m)$.

The STFT of a signal is therefore usually represented as a two-dimensional image. Each column (m) of the image corresponds to a time window of a signal, while the rows (k) indicate how much of each frequency is present at a given time. It is therefore a bit like the sheet music score for a piano composition, in which the left-to-right axis of the music staff corresponds to time, while the markings up and down the staff correspond to the notes (i.e., frequencies) that are played at the given time.

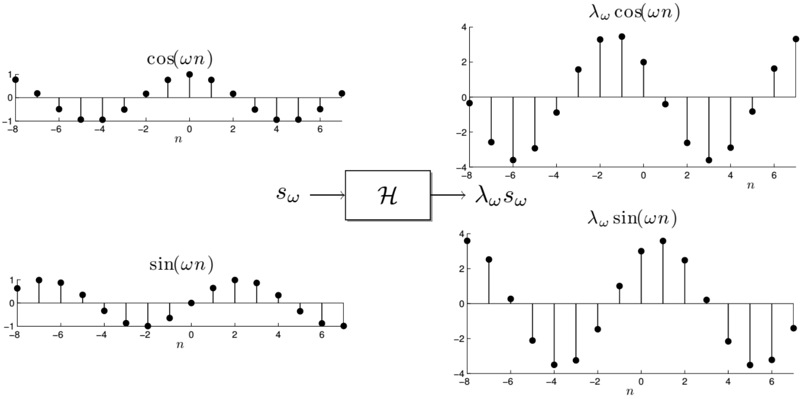

time m

Eigenanalysis of LTI Systems (Infinite-Length Signals)

Complex harmonic sinusoids are eigenvectors of discrete-time infinite length LTI systems. When given as inputs to any LTI system, the output is simply a scaled version of the input.

Eigenvectors of LTI Systems (infinite-length)

For a system H operating on infinite-length discrete-time signals, an eigenvector is a signal which, when passing through the system, is modified only by a scalar factor. So suppose s[n] is an eigenvector to system H, then $H\{s[n]\}=\lambda ss[n]$, where λs is a complex number. It is a remarkable property that all LTI systems have a set of eigenvectors, and furthermore that these eigenvectors are of the form $ej\omega n$. This means that, whatever ω may be, if $ej\omega n$ is input into any LTI system, the output will simply be a scaled version of the input (the amount of scaling depending on the nature of the system, of course):



Proving this special property of LTI systems is straightforward; we simply express the output in terms of the complex harmonic sinusoid input ($s_\omega[n]=e^{j\omega n}$) and system impulse response ($h[n]$) through the convolution sum, and simplify:

$$s_\omega[n]*h[n]=\sum_{m=-\infty}^{\infty} s_\omega[n-m]\,h[m]=\sum_{m=-\infty}^{\infty} e^{j\omega(n-m)}h[m]=\sum_{m=-\infty}^{\infty} e^{j\omega n}e^{-j\omega m}h[m]=\left(\sum_{m=-\infty}^{\infty} h[m]e^{-j\omega m}\right)e^{j\omega n}=\lambda_\omega s_\omega[n]$$

The Frequency Response of an LTI System

Take a close look at the value λ_ω above. It is intimately related to the system, as we would expect, by the system's impulse response:

$$\lambda_\omega=\sum_{m=-\infty}^{\infty} h[m]e^{-j\omega m}.$$

This is simply the DTFT of the impulse response! It is so important, we give it a special notation $H(\omega)$, and a special name as well. $H(\omega)$ is called the **frequency response**, for it explains how the system responds (i.e., in what way it scales) to particular input frequencies ω. This is also apparent by seeing the DTFT as an inner product of $h[n]$ with the sinusoidal $e-j\omega m$. The value of the inner product (which is the value $H(\omega)$ grows or shrink in ways corresponding to the similarity of $h[n]$ and $e-j\omega m$).

Although we cannot display the system matrix involved (as it is infinite in length), the DTFT diagonalizes the system H for infinite length signals, just as the DFT does with finite length ones. Expressing this diagonalization in terms of the DTFTs of the system, we have that if $y[n]=h[n]*x[n]$, then $Y(\omega)=H(\omega)X(\omega)$.

The Discrete-Time Fourier Transform

It is a significant concept in signal processing that there is more than one way to express the information in a given signal. One way is by defining the signal in terms of the value it has at any given time n, thus the notation x[n]. But is also possible to express a signal in terms of how it is be a combination of signals from a particular signal set. We could express the information in the signal by noting how much of each signal in the signal set contributes in the combination.

The DTFT of an infinite-length signal can be thought of as the limit of adding an infinite number of zeroes before and after a finite-length signal.

Definition of the DTFT

The **discrete-time Fourier transform**, or **DTFT**, is a common way to express infinite-length discrete-time signals in another way. As noted above, the signal x[n] can be understood as an entity as having a particular value for every time n in the range $-\infty < n < \infty$. But it also can be understood in this way:

$$x [n] = 1 \; 2 \pi \int - \pi \; \pi X (\omega) e \; j \omega n \; d \omega$$

x[n] is expressed in terms of how much of each signal $ej\omega n$ contributes to the integral which composes it. The weighting of each signal is according to the function X(ω). The values of that function are found according to this formula:

$X(\omega)=\Sigma n=-\infty\,\infty\, x[n]e-j\omega n.$

This formula for X(ω) is known as the DTFT of x[n], while that first integral formula is known as the inverse DTFT.

Interpretations of the DTFT

We have given the formulas for the DTFT and the inverse DTFT:

- $X(\omega)=\Sigma n=-\infty\,\infty\, x[n]e-j\omega n$
- $x[n]=1\,2\pi\int-\pi\,\pi\, X(\omega)e j\omega n\, d\omega$

But what do these formulas mean? There are a few ways of understanding them. One is to say that the two formulas are expressing the same entity, but in different domains: x[n] is the *time-domain* representation of the signal, whereas X(ω) is the *frequency-domain* representation of it. In that sense, it is like two different language translations of the same book (this analogy is not perfect, for translations are not exactly one-to-one; there are many English-language translations of the one Spanish book *Don Quixote*, but there is only one frequency-domain representation X(ω) for a time-domain signal x[n]). A second way to understand the DTFT is in terms of the concepts of analysis and synthesis. The DTFT formula gives us a frequency analysis of the signal x[n]. The signal X(ω) analyzes x[n] from a different point of view; it tells us the frequency content of x[n], whether it is composed of only low frequencies, or just high ones, or whatever the case may be. This is because every value of X(ω) is simply the inner product of x[n] with ejωn; it tells us the "strength" of ejωn in the signal x[n]. The inverse DTFT then synthesizes, or composes, x[n] based upon its weight at each different frequency.

Relationship between the DTFT and the DFT

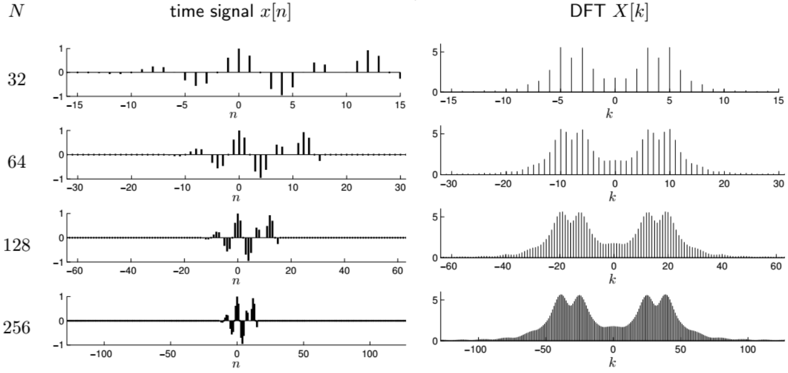
The DTFT is not the only Fourier transform for discrete-time signals. While the DTFT operates on infinite-length signals, the DFT (discrete Fourier transform) is a tool for frequency analysis of finite-length signals. A finite-length signal (say, of length N) x[n] has an N length frequency representation X[k], according to these formulas:

X[k] = Σn = 0N − 1x[n]e − j2πNknx[n] = 1NΣk = 0N − 1X[k]ej2πNkn.

As x[n] and X[k] are N-periodic, we can also represent them in this way, shifted about n and k at the origin:

X[k] = Σn = − N2N2 − 1x[n]e − j2πNknx[n] = 1NΣk = − N2N2 − 1X[k]ej2πNkn.

We could develop the DTFT by seeing what happens to the DFT as N gets larger and larger. As N approaches ∞, so also do n and k run from $-\infty$ to ∞. Consider also what happens to $e-j2\pi Nkn$. As N gets infinitely large, the exponent term $2\pi Nk$ becomes a continuous value, running from $-\pi$ to π, and we can call that value ω. We have then that the DFT sum becomes: $X(\omega)=\Sigma n=-\infty\infty x[n]e-j\omega n$ which is the definition of the DTFT. Similarly, on the synthesis side of the transform, the DFT "sum" will not run over a discrete number of values k, but rather become an integral over the continuous variable ω: $x[n]=\int-\pi\pi X(\omega)ej\omega nd\omega 2\pi$ which is our inverse DTFT. Graphically, we can see how increasing the length of a signal by adding more and more zeros on either side of it results in the signal's DFT (which is expressed in terms of k) appearing to be more and more like a continuous function (expressed in terms of ω):



Discrete-Time Fourier Transform Examples

The DTFT and inverse DTFT are defined as follows:

$X (\omega) \quad = \quad \Sigma n = - \infty \infty x [n] e - j \omega n , \quad - \pi \le \omega < \pi x [n] \quad = \quad \int - \pi \pi X (\omega) e j \omega n d \omega 2 \pi , \qquad \infty < n < \infty$

Let's work out some examples of DTFT and inverse DTFT calculations.

Impulse Response of an Ideal Lowpass Filter

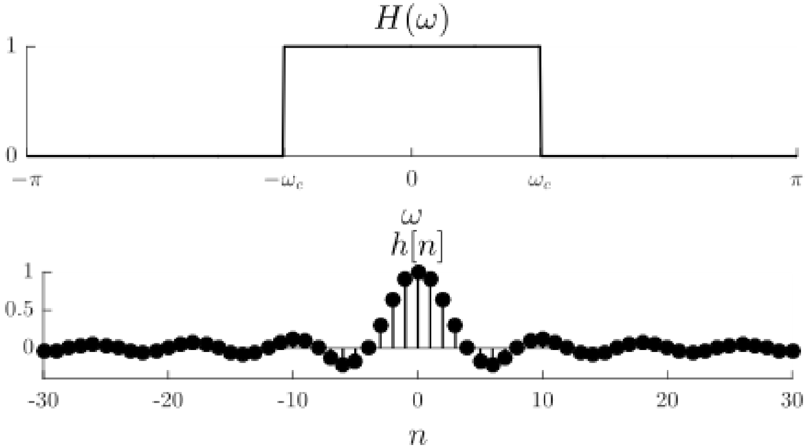
We'll start with an ideal lowpass filter. From its frequency response we can see that it blocks all incoming frequencies having a magnitude greater than $|\omega c|$:

$H(\omega)=\{1-\omega c\le\omega\le\omega c 0 \text{otherwise}.$

In order to find the impulse response for such a filter, we will perform an inverse DTFT on the frequency response:

$h [n] = \int - \pi \pi H (\omega) e j \omega n d \omega 2 \pi = 1 2 \pi \int - \omega c \omega c 1 e j \omega n d \omega = 1 2 \pi [1 j n e j \omega n] - \omega c \omega c = 1 \pi n (e j \omega c n - e j \omega c n) 2 j = \sin (\omega c n) \pi n$

This impulse response is a form of the sinc function, and is plotted below.

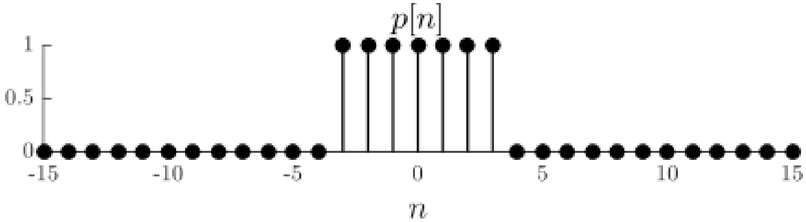


A symmetrical pulse, with $M=3$.A digital sinc function.

DTFT of a Moving Average System

Consider now a moving average system, where the system output at a time n is the average of M input values, symmetric about n. The impulse response of such a system is a pulse function:

$p[n]=\{1-M\leq n\leq M0\text{otherwise}.$

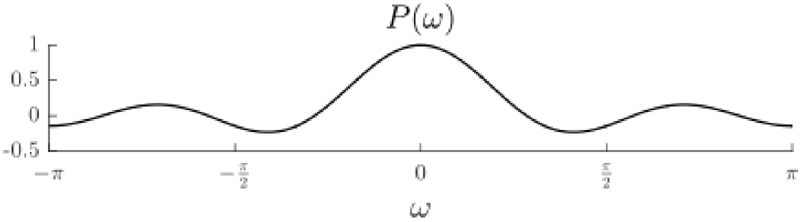


Suppose we would like to find the frequency response of this system. We merely need to take the

DTFT of the impulse response:

P(ω) = Σn= − ∞ ∞p[n]e−jωn = Σn= −MMe −jωn = Σn= −MM(e−jω)n = ejωM−e−jω(M +1)1−e−jω = ejωM−e−jω(M+1)1−e−jω = e −jω/2(ejω2M+12−e−jω2M+12)e −jω/2(ejω/2−e−jω/2) = 2jsin(ω2M +12)2jsin(ω2) = sin(2M+12ω)sin(ω2).

This frequency-domain representation of the time-domain pulse is known as a digital sinc:

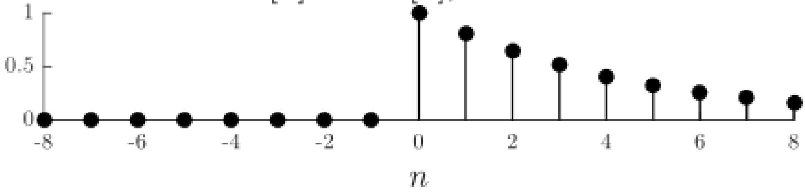


The digital sinc function is similar to a regular sinc function, except that rather than perpetually decaying, it is 2π periodic. When we compare this to the first example, the impulse response of an ideal lowpass filter, we see an interesting correspondence. Pulses in one domain (either time or frequency) have sinc-like representations in the alternate domain.

The impulse response of a recursive average system. The magnitude of the frequency response of a one-sided exponential.

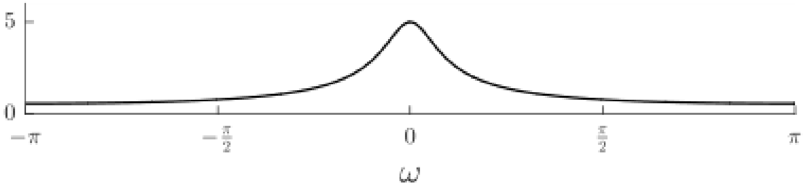
DTFT of a One-Sided Exponential

Having found the frequency response of a moving average system, let's now do the same for a recursive average system. The input/output relationship of such a system can be expressed as $y[n]=x[n]+\alpha y[n-1]$, where $|\alpha|<1$. The impulse response for such a system is $h[n]=\alpha n u[n]$:


$h[n]=\alpha^n u[n], \quad \alpha=0.8$

We'll now take the DTFT of this impulse response:

$H(\omega) = \Sigma n=-\infty\infty h[n]e-j\omega n = \Sigma n=0\infty \alpha ne -j\omega n = \Sigma n=0\infty(\alpha e-j\omega)n = 11-\alpha e-j\omega.$


$|H(\omega)|$

The Discrete-Time Fourier Transform of a Sinusoid

Recall the definition of the DTFT of a signal x[n]:

$X(\omega) = \Sigma n = -\infty \infty x[n]e - j\omega n.$

Let's see what the DTFT of a complex exponential signal $ej\omega 0n$:

$X(\omega 0) = \Sigma n = -\infty \infty x[n]e - j\omega n = \Sigma n = -\infty \infty ej\omega 0ne - j\omega n = \Sigma n = -\infty \infty e - j(\omega - \omega 0)n.$

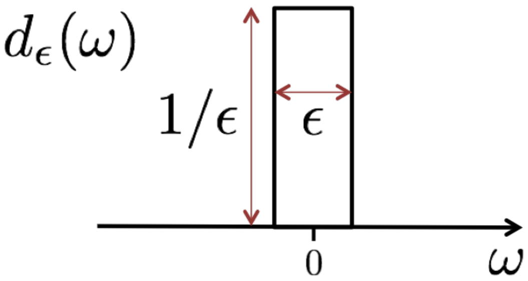
The nature of that sum is not immediately clear. When $\omega = \omega 0$, then the sum becomes unbounded:

$\Sigma n = -\infty \infty 1 = \infty.$

But when $\omega \neq \omega 0$... what does an infinite sum for a sinusoid mean?

The Dirac Delta Function

To find the DTFT of a sinusoid, it will to be necessary to consider another function, the dirac delta function. Suppose we have a function $d\epsilon(\omega)$:



Note how the non-zero portion of this function has a width of ϵ and a height of 1ϵ. The area underneath the function is therefore always 1, no matter what ϵ may be. We will let the value of ϵ get smaller and smaller. As it approaches zero, the function will become taller and narrower, until at the limiting case it is "infinitely" tall and "infinitesimally" narrow. We will denote the function in this limiting case $\delta(\omega)$. But it will still have an area of 1, thus

$$\int \delta(\omega) d\omega = 1.$$

Suppose we scale the $\delta(\omega)$ by another function $X(\omega)$. Since $\delta(\omega)$ is non-zero at all values other than at $\omega = 0$, scaling $\delta(\omega)$ by $X(\omega)$ within the integral will yield the following:

$$\int X(\omega) \delta(\omega) d\omega = X(0).$$

What if the integrand has $\delta(\omega - \omega 0)$ and is scaled by $X(\omega)$. In this case the integrand is only nonzero at $\omega = \omega 0$, thus:

$$\int X(\omega) \delta(\omega - \omega 0) d\omega = X(\omega 0).$$

The $\delta(\omega)$ "function" we have derived is known as the **Dirac delta function**. It is not technically a function (it is a generalized function, or distribution), but for our purposes we can treat it as a function with infinite height, infinitesimal narrowness, and an area of 1 at $\omega=0$.

The DTFT of a Sinusoid

Having introduced the Dirac delta function, let's see what happens if we attempt to find the inverse DTFT of $2\pi\delta(\omega-\omega 0)$:

$$\int -\pi\pi 2\pi\delta(\omega-\omega 0)ej\omega nd\omega 2\pi \ = \ ej\omega 0n.$$

The inverse DTFT of a dirac delta function is a complex sinusoid, which means that the DTFT of a complex sinusoid is a dirac delta:

$$ej\omega 0n \ \longleftrightarrow \text{DTFT} \ 2\pi\delta(\omega-\omega 0).$$

Via Euler's formula, we use the above to also find the DTFT of discrete-time cosines and sines:

$$\cos(\omega 0n) \ \longleftrightarrow \text{DTFT} \ \pi\delta(\omega-\omega 0)+\pi\delta(\omega+\omega 0)\sin(\omega 0n) \ \longleftrightarrow \text{DTFT} \ \pi j\delta(\omega-\omega 0)+\pi j\delta(\omega+\omega 0).$$

Discrete-Time Fourier Transform Properties

Recall the definition of the DTFT and the inverse DTFT:

$X(\omega) = \Sigma n=-\infty \infty x[n]e$
$-j\omega n, \quad -\pi\leq\omega<\pi x[n] = \int$
$-\pi\pi X(\omega)ej\omega nd\omega 2\pi, \quad \infty<n<\infty$.

A signal and its corresponding DTFT is referred to as a DTFT signal pair:

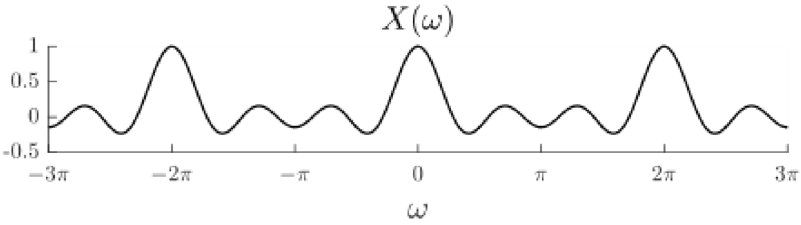
$x[n] \leftrightarrow DTFT \; X(\omega)$

We'll now take a look at some properties of the DTFT.
The 2π periodicity of the DTFT.

DTFT Periodicity

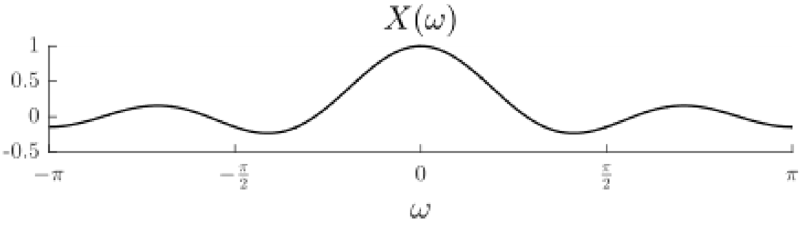
Although the DTFT of a signal is typically defined only on a 2π interval, it is nevertheless straightforward to show that the DTFT is actually 2π periodic:

$X(\omega+2\pi k) = \Sigma n=-\infty \infty x[n]e-j(\omega$
$+2\pi k)n = \Sigma n=-\infty \infty x[n]e-j\omega ne-j2\pi kn = X(\omega)$.



DTFT Frequencies

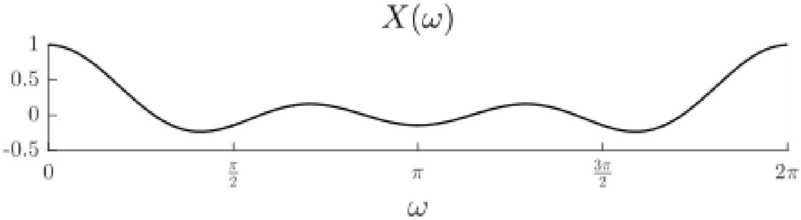
Because X(ω) is essentially the inner product of a signal x[n] with the signal ejωn, we can say that X(ω) tells us how strongly the signal ejωn appears in x[n]. X(ω), then, is a measure of the "frequency content" of the signal x[n]. Consider the plot below of the DTFT of some signal x[n]:



This plot shows us that the signal x[n] has a significant amount of low-frequency content (frequencies around ω = 0), and less high-frequency content (frequencies around ω = ±π -- remember that the DTFT is 2π periodic).

As the DTFT is 2π periodic, all the information in a DTFT is contained in any 2π length section. However, there are only two intervals that are typically used when representing a DFTF of a signal.

The first is to use the interval of $-\pi$ to π, as above. In such an interval, the low frequencies are at the center, with the high frequencies at the edges. The other interval ranges from 0 to 2π. The low frequencies are at the extremities of the plot, with the highest frequency in the center:



The DTFT and Time Shifts

If a signal is shifted in time, what effect might this have on its DTFT? Supposing $x[n]$ and $X(\omega)$ are a DTFT pair, we have that

$$x[n-m] \xleftrightarrow{\text{DTFT}} e^{-j\omega m}X(\omega).$$

So shifting a signal in time corresponds to a modulation (multiplication by a complex sinusoid) in frequency. We can use the DTFT formula to prove this relationship, by way of a change of variables $r=n-m$:

$$\sum_{n=-\infty}^{\infty} x[n-m]e^{-j\omega n} = \sum_{r=-\infty}^{\infty} x[r]e^{-j\omega(r+m)} = \sum_{r=-\infty}^{\infty} x[r]e^{-j\omega r}e^{-j\omega m} = e^{-j\omega m}\sum_{r=-\infty}^{\infty} x[r]e^{-j\omega r} = e^{-j\omega m}X(\omega)$$

The DTFT and Time Modulation

We saw above how a shift in time corresponds to modulation in frequency. What do you suppose happens when a signal is modulated in time? If you guessed that it is shifted in frequency, you're right! If a signal $x[n]$ has a DTFT of $X(\omega)$, then we have this DTFT pair:

$$ej\omega 0nx[n] \longleftrightarrow \mathrm{DTFT}\ X(\omega-\omega 0).$$

Below is the proof:

$$\Sigma\ n = \infty\ \infty\ e\ j\ \omega\ 0\ n\ x\ [\ n\ ]\ e\ -\ j\ \omega\ n\ \Sigma\ n = -\ \infty\ \infty\ x\ [\ n\ ]\ e\ -\ j\ (\ \omega\ -\ \omega\ 0\ )\ n = X\ (\ \omega\ -\ \omega\ 0\ )$$

The DTFT and Convolution

Suppose that the impulse response of an LTI system is $h[n]$, the input to the system is $x[n]$, and the output is $y[n]$. Because the system is LTI, these three signals have a special relationship:

$$y[n] = x[n]*h[n] = \Sigma m=-\infty\ \infty h[n-m]x[m].$$

The output $y[n]$ is the convolution of $x[n]$ with $h[n]$. Just as with the other DTFT properties, it turns out there is also a relationship in the frequency domain. Consider the DTFT of each of those signals; call them H(ω), X(ω), and Y(ω). The convolution of the signals x and h in time corresponds to the multiplication of their DTFTs in frequency:

$$y[n] = x[n] * h[n] \xleftrightarrow{\text{DTFT}} Y(\omega) = X(\omega) H(\omega).$$

For the proof, we take the DTFT of y[n], using a change of variables along the way:

$$Y(\omega) = \sum_{n=-\infty}^{\infty} y[n] = \sum_{n=-\infty}^{\infty} (x[n] * h[n]) e^{-j\omega n} = \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} x[m] h[n-m] e^{-j\omega n} = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x[m] h[n-m] e^{-j\omega m} e^{-j\omega (n-m)} = \sum_{m=-\infty}^{\infty} x[m] e^{-j\omega m} \left(\sum_{n=-\infty}^{\infty} h[n-m] e^{-j\omega (n-m)} \right) \text{Let } v = n-m = \sum_{m=-\infty}^{\infty} x[m] e^{-j\omega m} \left(\sum_{v=-\infty-m}^{\infty-m} h[v] e^{-j\omega (v)} \right) = \sum_{m=-\infty}^{\infty} x[m] e^{-j\omega m} \left(\sum_{v=-\infty}^{\infty} h[v] e^{-j\omega (v)} \right) = \sum_{m=-\infty}^{\infty} x[m] e^{-j\omega m} (H(\omega)) = H(\omega) \sum_{m=-\infty}^{\infty} x[m] e^{-j\omega m} = X(\omega) H(\omega)$$

This relationship is very important. It gives insight, showing us how LTI systems modify the frequencies of input signals. It is also useful, because it gives us an alternative way of finding the output of a system. We could take the DTFTs of the input and impulse response, multiply them together, and then take the inverse DTFT of the result to find the output. There are some cases where this process might be easier than finding the convolution sum.

The DTFT and Linearity

Since the DTFT is an infinite sum, it should come as no surprise that it is a linear operator. Nevertheless, it is a helpful property to know. Suppose we have the following DTFT pairs:

$x1[n] \longleftrightarrow \mathrm{DTFT}\ X1(\omega) \qquad x2[n] \longleftrightarrow \mathrm{DTFT}\ X2(\omega).$

Then by the linearity of the DTFT we have that, for any constants $\alpha 1$ and $\alpha 2$:

$$\alpha 1 x 1 [n] + \alpha 2 x [2] \ \longleftrightarrow \mathrm{D\,T\,F\,T} \ \ \alpha 1 X 1 (\omega) + \alpha 2 X 2 (\omega)$$

DTFT Symmetry

Looking at the plots of the DTFTs above, or perhaps of those you have computed on your own, you may notice that they are often symmetrical. It so happens that the DTFTs of real-valued signals always have a kind of symmetry, but there are other symmetry relationships as well. For example, signals that are purely imaginary and odd have DTFTs that are purely real and odd. These types of symmetry are a result of a property of the complex exponentials which build up the DTFTs. Any signal of the form ejωn is conjugate symmetric, meaning that its real part is even and its imaginary part is odd. Additionally, for conjugate symmetric signals, their magnitude is even and their phase is odd.

As a result of the symmetry of the signals which compose a DTFT, the DTFTs of certain signals also have symmetries, according to the table below:

$x[n]$	$X(\omega)$	$\mathrm{Re}\,(X(\omega))$	$\mathrm{Im}\,(X(\omega))$	$\lvert X(\omega)\rvert$	$\angle X(\omega)$
real	conjugate symmetric: $X(-\omega) = X(\omega)^*$	even	odd	even	odd
real and even	real and even	even	zero	even	$0, \pi$
real and odd	imaginary and odd	zero	odd	even	$\pm \pi 2$
imaginary	conjugate odd symmetric $X(-\omega) = -X$	odd	even	even	odd

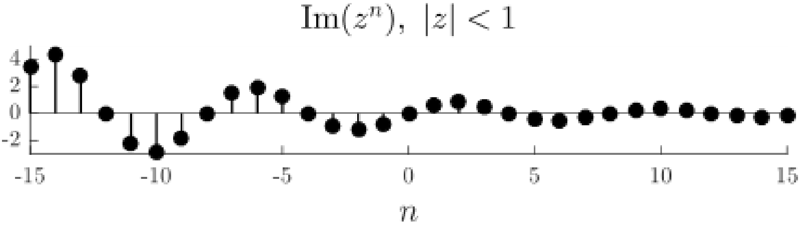
(ω) *

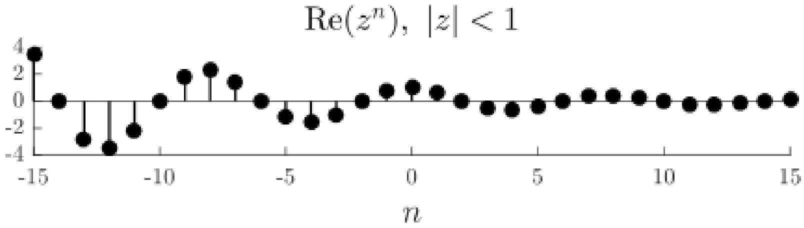
imaginary and even	imaginary and even	zero	even	even	± π2
imaginary and odd	real and odd		zero	even	0 , π

Generalizing the DTFT

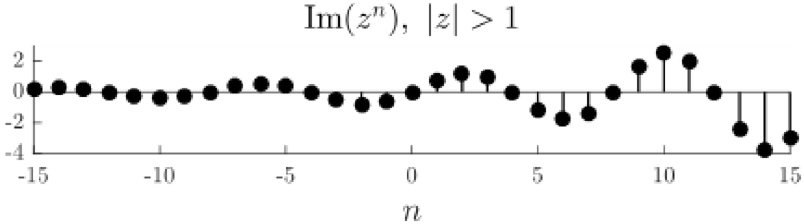
We have seen that the DTFT is a useful tool for representing and analyzing discrete-time infinite length signals and LTI systems. With the DTFT, signals can be represented in terms of their frequency composition. Because complex harmonic sinusoids are eigenvectors of LTI systems, the DTFT of a system's impulse response gives us the system's frequency response (it tells us how the system modifies input signals according to their composite frequencies).

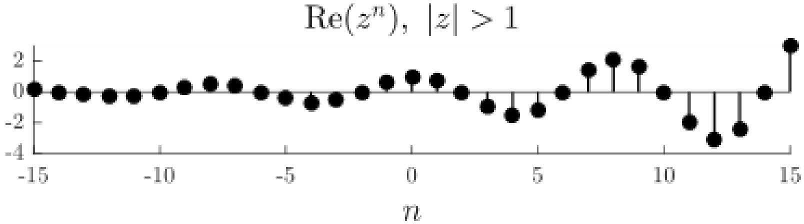
The backbone of the DTFT is the complex harmonic sinusoid: ejωn. This type of signal is actually a specific example of a more broader class of signals zn, where z is a complex number. Signals in this class are called complex exponentials. As zn = |z|nej(∠z)n, we see that complex exponentials are simply complex harmonic sinusoids, with exponential function envelopes. When |z| < 1, they decay in time:





When |z|>1, they grow in time:





And when |z|=1, they are of course simply complex harmonic sinusoids of the form ejωn.

Since complex harmonic sinusoids have the special property that they are eigenvectors of LTI systems, we may ask if the more general signal class of complex exponentials does, as well. To test, we will find the an LTI system's output (i.e., the result of the convolution sum) to some generic complex exponential zn:

y[n]=zn∗h[n]=Σm=−∞∞x[n−m]h[n]=Σm=−∞∞zn−mh[n]=Σm=−∞∞znz−mh[n]=znΣm=−∞∞z−mh[n]=(Σm=−∞∞z

$-mh[n])zn=H(z)zn$, $H(z)=\Sigma m=-\infty\ \infty z-mh[n]$.

If we input a complex exponential into some LTI system, then indeed the output is simply a scaled version of that same exponential. So complex exponentials are also eigenvectors of LTI systems. Now, the amount a given LTI system scales a complex exponential zn is very important, so much so that it has a special name--it is the system's **transfer function** (it tells us how the system "transfers" the input into the output)--and a special label, $H(z)$.

This transfer function of LTI systems, $H(z)$, is clearly a generalization of the frequency response $H(\omega)$. The frequency response indicated how an LTI system H scales input complex sinusoids $ej\omega$, and likewise the transfer function $H(z)$ tells us how the system scales input complex exponentials zn. As $ej\omega n=zn|z=ej\omega$ it follows that $H(\omega)=H(z)|z=ej\omega$. So it would seem that the DTFT of a function (for $H(\omega)$ is the DTFT of $h[n]$) can be generalized to be a broader kind of transform, which is represented by $H(z)$. This broader class of transform is called the z-transform.

The z-Transform

Let $x[n]$ be a discrete-time infinite length signal. Then $X(z)=\Sigma n=-\infty\ \infty x[n]z-n$ is defined to be the bilateral **z-transform** of x[n] (the unilateral z-transform sums only from $n=0$ to ∞; unless otherwise stated, we assume the transform is bilateral).

There are a few things we can note immediately about the z-transform. First, unlike either transform we have considered (the DFT and DTFT), it is a complex function of a *complex* variable. Second, and like the DTFT (but not the DFT), the z-transform may not necessarily exist for all z and/or all x[n]. And third, like both other transforms, the z-transform diagonalizes LTI systems. By this, we mean that if a signal x[n] is expressed in terms of its z-transform, then the z-transform of the output at various z is simply the pointwise multiplication of the input's transform with the transform of the impulse response (i.e., the transfer function: $Y(z)=H(z)X(z)$).

The Existence of the z-Transform

Recall the definition of the z-transform for some discrete-time infinite length signal $x[n]$:

$$X(z)=\sum_{n=-\infty}^{\infty} x[n] z^{-n}.$$

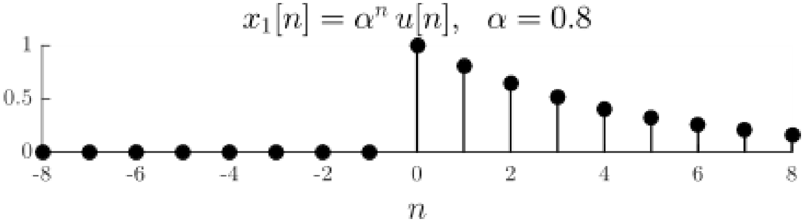
Given that this is an *infinite* sum (unlike, say, the sum of N terms in a DFT), it will not necessarily converge for all $x[n]$ and/or all z. For some $x[n]$, it will converge for all values of z. For example, if $x[n]=\delta[n]$, then:

$$X(z)=\sum_{n=-\infty}^{\infty} \delta[n] z^{-n}=1.$$

For some $x[n]$, the sum will not converge for any z. For example, if $x[n]=1$, there is no z such that the sum converges.

The Region of Convergence

So for some $x[n]$ (like $x[n]=\delta[n]$) the z-transform sum will converge for *all* z, and for some $x[n]$ (like $x[n]=1$) the z-transform will not converge for *any* z. As you might suspect, there are some $x[n]$ for which the sum converges for *some* z. These values of z, those for which the z-transform exists, are known as the z-transform's **region of convergence**. Consider, for example the z-transform for the signal $x1[n]=\alpha nu[n]$, $\alpha=.8$:

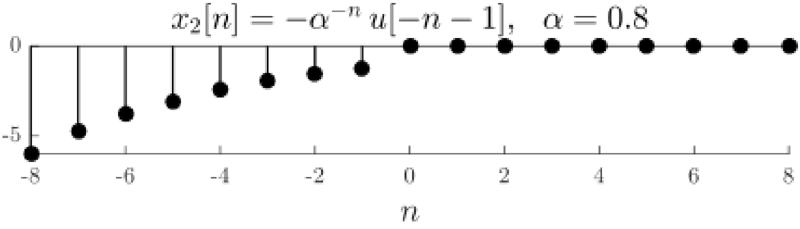


$X1(z)=\Sigma n=-\infty\infty x1[n]z-n=\Sigma n=0\infty\alpha nz-n=\Sigma n=0\infty(\alpha z-1)n=\{11-\alpha z-1|\alpha z-1|<1$undefinedelse.

So, the z-transform X1(z) only exists for $|\alpha z-1|<1$, or put another way, for $|z|>\alpha$; it is not defined for other values of z. This should not be too terribly shocking, for we are familiar with other functions that only exist for certain values, e.g. logx only for $x>0$, or 1x only for $x\neq 0$.

However, there is a difference between the z-transform and its region of convergence and functions like logx or 1x. Where the defined regions of existence of the values of logx or 1x are quite clear, there is nothing intrinsic about the function $zz-\alpha$ to suggest it only exists for $|z|>\alpha$ (of course it is apparent that z cannot equal α). Why in the world would it be a problem if $z=\alpha/2$, for then we would simply have $\alpha/2\alpha/2-\alpha=-1$, right? The reason is that the function was the simplification of the *sum*, a simplification that existed only for certain z.

To see why that is the case, consider another signal related to the one we just considered. Let x2[n] = −αnu[−n−1] , α=.8:



We will now consider its z-transform:

X 2 (z) = Σ n = − ∞ ∞ x 2 [n] z − n = Σ n = − ∞ − 1 − α n z − n = − Σ n = 1 ∞ α − n z n = − Σ n = 0 ∞ (α − 1 z) n + 1 = { − 1 1 − α − 1 z + 1 | α − 1 z | < 1 undefined else = { − 1 1 − α − 1 z + 1 − α − 1 z 1 − α − 1 z | α − 1 z | < 1 undefined else = { 1 1 − α z − 1 | α − 1 z | < 1 undefined else

So for x2[n], its z-transform is defined only for |α −1z|<1, or equivalently, |z|<α, and for those values it is 11−αz−1. Now we see another reason why the region of convergence is so important; it is not merely added information about the z-transform, it is intrinsic to its very definition. The z-transforms for x1[n] and x2[n] are completely different, which is to be expected as those are completely different signals. However, the only difference between the transforms is the region of convergence:

$$x_1[n] = \alpha^n u[n] \leftrightarrow X_1(z) = \begin{cases} \frac{1}{1-\alpha z^{-1}} & |z| > \alpha \\ \text{undefined} & \text{else} \end{cases} \quad x_2[n] = -\alpha^n u[-n-1] \leftrightarrow X_2(z) = \begin{cases} \frac{1}{1-\alpha z^{-1}} & |z| < \alpha \\ \text{undefined} & \text{else} \end{cases}$$

Types of ROCs

We have seen a few different types of ROCs for the z-transforms of various signals. If the z-transform for a discrete-time signal exists, then the region of convergence will fall into one of these three categories:

- If the signal has a finite non-zero duration, then the region of convergence will be the entire z-plane, with the possible exception of $z=\infty$ (if the signal has nonzero values at time indices less than zero) or $z=0$ (if the signal has nonzero values at time indices greater than zero).
- If the signal is of infinite duration as n approaches ∞ (but not $-\infty$), then the region of convergence will extend outside some disk (e.g., $|z|>1$)
- If the signal is of infinite duration as n approaches $-\infty$ (but not ∞), then the region of convergence will be inside some disk (e.g., $|z|<1$)

- If the signal is of infinite duration as n approaches both ∞ and $-\infty$, then the region of convergence--if it exists at all--will be an annulus (e.g., $1 < |z| < 2$)

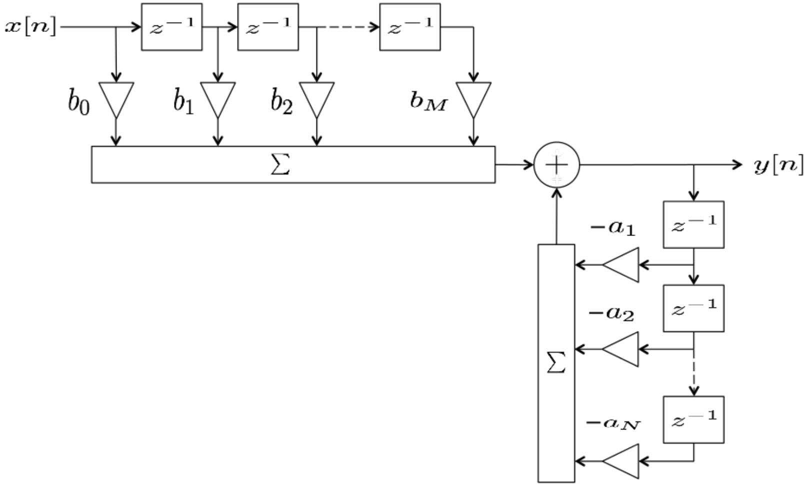
The Transfer Function of Discrete-Time LTI Systems

LTI Systems in the Time Domain

One of the most general ways of defining an LTI system is by expressing its output at some time n as a weighted sum of values of the input and output at a finite number various other times. If all those times are at or prior to n, such a system is causal. In terms of a difference equation, the system definition would look something like this:

$$y[n] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] + \cdots + b_M x[n-M] - a_1 y[n-1] - a_2 y[n-2] + \cdots - a_N y[n-N]$$

This kind of system can also be displayed as a block diagram:



In the block diagram, the $z-1$ blocks represent time delays, the triangles represent multiplications, and the Σ blocks represent summations of the signals being input into them. The output at time n is a weighted sum of the previous M input values (along with the current value) and previous N output values.

LTI Systems in the z-Domain

Consider again the input/output relationship of a causal LTI system:

$$y[n] \ = \ b0x[n]+b1x[n-1]+b2x[n-2]+\cdots +bMx[n-M]-a1y[n-1]-a2y[n-2]+\cdots-aNy[n-N].$$

We can take the z-transform of this whole equation, bearing in mind that the transform is a linear operation and that delays of n0 in time correspond to multiplications by $z-n0$ in the z-domain. That would leave us with this:

$$Y(z)= \ b0X(z)+b1z-1X(z)+b2z-2X(z)+\cdots bMz-MX(z)-a1z-1Y(z)-a2z-2Y(z)-\cdots-aNz-NY(z).$$

From there we can do a little rearranging:

$$Y(z)=b0X(z)+b1z-1X(z)+b2z-2X(z)+\cdots bMz$$

−MX(z)−a1z−1Y(z)−a2z−2Y(z)−⋯−aNz
−NY(z)Y(z)+a1z−1Y(z)a2z−2Y(z)+⋯+aNz
−NY(z)=b0X(z)+b1z−1X(z)+b2z−2X(z)+⋯bMz
−MX(z)Y(z)(1+a1z−1+a2z−2+⋯+aNz
−N)=X(z)(b0+b1z−1+b2z−2+⋯bMz
−M)H(z)=Y(z)X(z)=b0+b1z−1+b2z−2+⋯bMz
−M1+a1z−1+a2z−2+⋯+aNz−N.

This H(z), which is a rational function in terms of z, is called the system's **transfer function**.
A system with two poles has three possible regions of convergence.Consider a system with two poles, as plotted here. One region of convergence could be a disc within the smallest magnitude pole. Another region of convergence could be an annulus between the two poles. A final region of convergence could be all values of z of greater magnitude than that of the largest pole.

The Transfer Function: Poles and Zeroes, ROC, and Stability

So we see that the transfer function for one of the most common classes of LTI systems--in which the output is a weighted sum of a finite number of past inputs and outputs--is a rational function in terms of z:

H(z)=b0+b1z−1+b2z−2+⋯bMz−M1+a1z
−1+a2z−2+⋯+aNz−N.

We can express this function in terms of positive powers of z by simple factoring, and then according to the fundamental theorem of algebra, we can represent the numerator and denominator polynomials with respect to their roots:

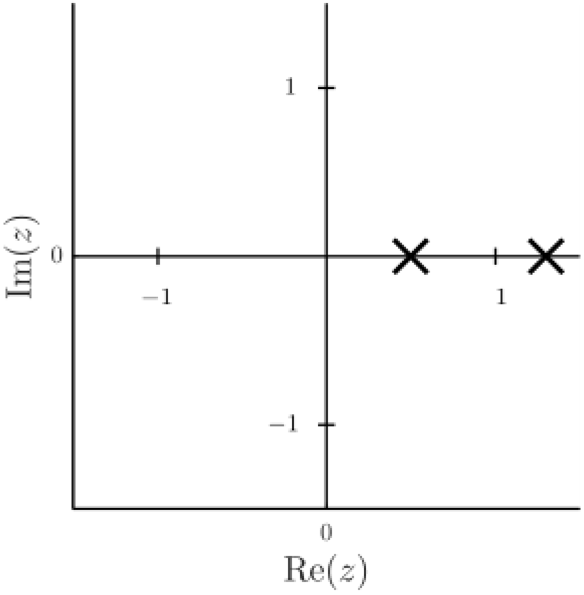
$H(z)=b0+b1z-1+b2z-2+\cdots bMz-M1+a1z-1+a2z-2+\cdots+aNz-N=z-M(b0zM+b1zM-1+b2zM-2+\cdots+bM)z-N(zN+a1zN-1+a2zN-2+\cdots+aN)=z-Mb0(z-\zeta1)(z-\zeta2)\cdots(z-\zeta M)z-N(z-p1)(z-p2)\cdots(z-pN).$

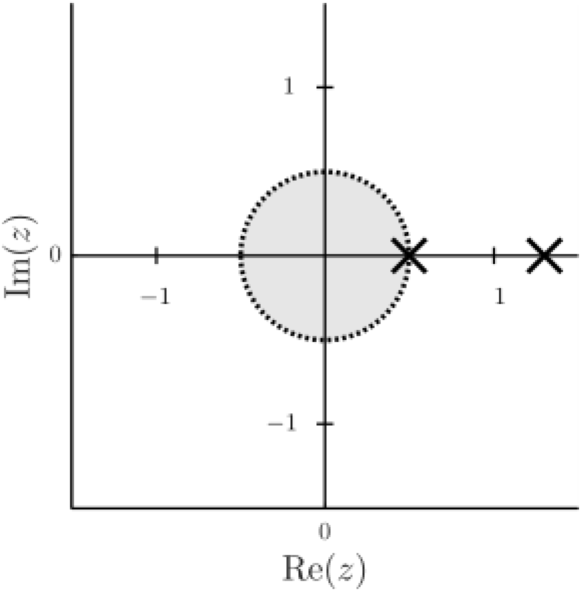
The values of z for which the numerator is zero are called **zeros**, and the values for which the denominator is zero are called **poles**.

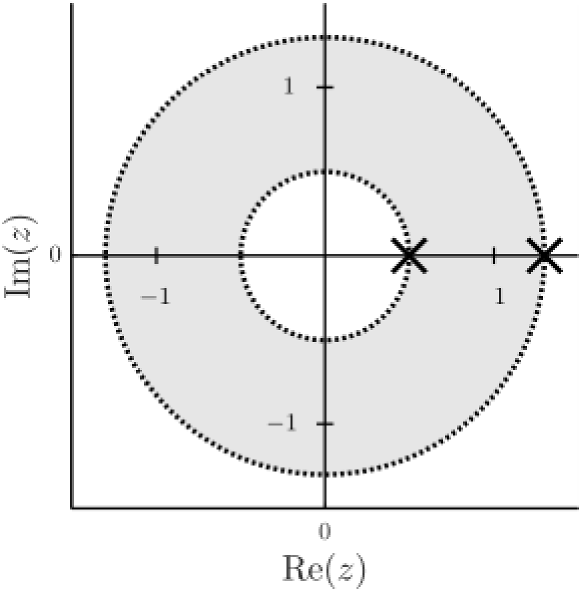
Now, there is a special reason the transfer function is given the notation H(z): it happens to be the z-transform of h[n], the impulse response of the system. Like all z-transforms, it has a region of convergence. We have seen previously that the ROC for a z-transform will be one of the following:

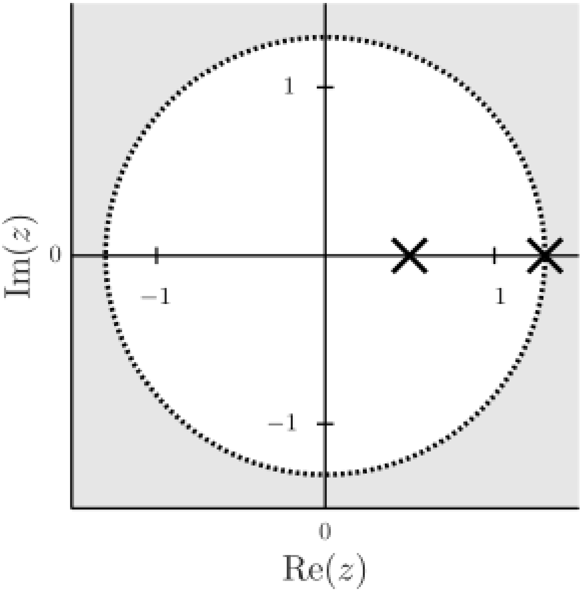
- no z (the z-transform does not converge for any z)
- the entire z plane (with the possible exception of $z=0$ or $z=\infty$)
- all z outside a certain disc (e.g., $|z|>1$)
- all z within a certain disc (e.g., $|z|<1$)
- z within an annulus (e.g., $1<|z|<2$)

Poles play a very significant role in the ROC for a transfer function; in fact, they completely define the ROC. The ROC of a z-transform cannot contain a pole (for the function would not exist at that point), which means that once we know the poles of a system, we know the potential ROCs. We say *potential* because there are a variety of possible ROCs for a transfer function with one or more poles; to be precise, if there are M distinct poles, then there will be M + 1 possible ROCs: the disc within the pole of smallest magnitude, the region outside the disc or radius equal to the largest pole, and then the M − 1 annuli of regions between adjacent poles. If, suppose, the system in question has two poles, then there will be three possible regions of convergence:




1
0
−1
1
−1
Im(z)
0
Re(z)

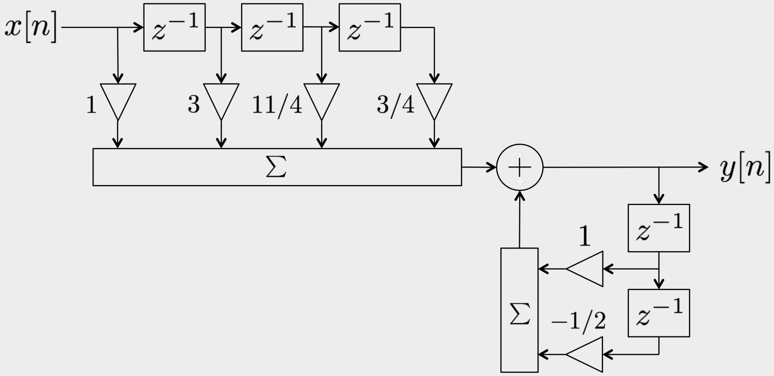

1
0
−1
1
−1
Im(z)
0
Re(z)



As we have also seen previously, each type of ROC corresponds to a particular kind of discrete-time signal. ROCs which are a disc correspond to signals with a non-zero duration extending to $n = -\infty$, those that are all z outside of a certain disc correspond to signals with a non-zero duration extending to $n = \infty$, and those that are an annulus correspond to signals whose non-zero duration extends both to $n = -\infty$ and $n = \infty$. For the class of discrete-time LTI system that we have been just considering, the impulse response is causal, and thus if it is infinite, will have a duration that extends towards $n = \infty$. Thus the transfer function of such a system has a ROC that extends outward from the outermost pole.

Finding the Transfer Funcion, Zeros and Poles, and ROC

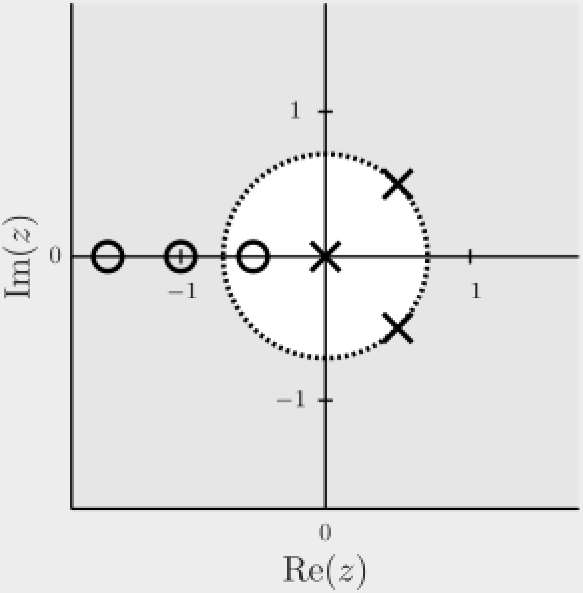
Consider a discrete-time system represented by this block diagram:



We would like to find its transfer function, along with its poles and zeros. The first step will be to express the input/output relationship in the time domain. From there we can take the z-transform of each side and find Y(z)X(z), which is by definition the transfer function:

y[n] = x[n] + 3x[n − 1] + 114x[n − 2] + 34x[n − 3] + y[n − 1] − 12y[n − 2]y[n] − y[n − 1] + 12y[n − 2] = x[n] + 3x[n − 1] + 114x[n − 2] + 34x[n − 3]Y(z) − z − 1Y(z) + 12z − 2Y(z) = X(z) + 3z − 1X(z) + 114z − 2X(z) + 34z − 3X(z)Y(z)(1 − z − 1 + 12z − 2) = X(z)(1 + 3z − 1 + 114z − 2 + 34z − 3)H(z) = Y(z)X(z) = 1 + 3z − 1 + 114z − 2 + 34z − 31 − z − 1 + 12z − 2 = z − 3(z3 + 3z2 + 114z + 34)z − 2(z2 − z + 12) = (z + 1)(z + 12)(z + 32)z(z − 12 − j2)(z − 12 + j2).

So, having found the transfer function and factored the numerator and denominator, we see readily that the zeros of the system are $z = -1/2,\ -1,\ -3/2$, and the poles for the system are $z = 0,\ 1/2 + j/2,\ 1/2 - j/2$. As the system is causal (from the block diagram we see the output is dependent only on present and/or past values of the input and output), the ROC will extend outward from the outermost pole. The two nonzero poles are equidistant from the origin, at a distance of 22, so the ROC is $|z| > 22$. A pole/zero plot, with the ROC shaded, is below:



From the Transfer Function to the

Frequency Response

We have seen that for LTI systems whose output is a weighted sum of a finite number of past inputs and outputs, the transfer function amounts to being a rational function of the form H(z) = b0 + b1z −1 + b2z − 2 + ⋯bMz − M1 + a1z − 1 + a2z − 2 + ⋯ + aNz − N, or equivalently, z − Mb0(z − ζ1)(z − ζ2)⋯ (z − ζM)z − N(z − p1)(z − p2)⋯(z − pN). One of the benefits in expressing the transfer function as the products of those single order polynomials is the elegant way it leads to understanding the magnitude of the transfer function:

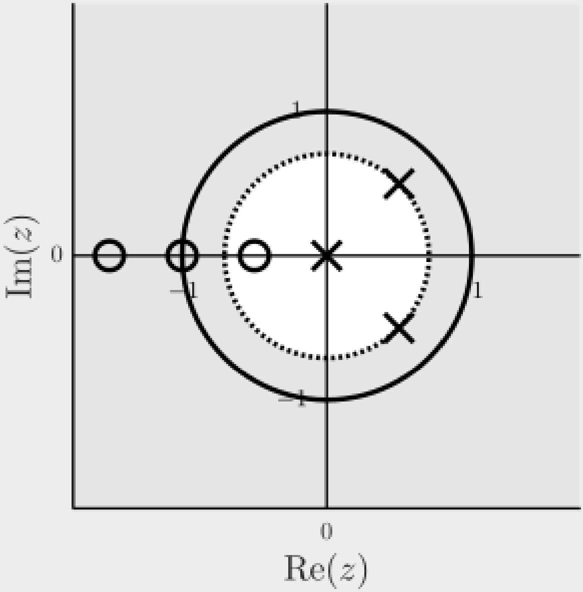
|H(z)| = |z − M||b0||(z − ζ1)||(z − ζ2)|⋯|(z − ζM)||z − N||(z − p1)||(z − p2)|⋯|(z − pN)|.

Note what terms like |z − p| represent. The magnitude is (Re (z) − Re (p))2 + (Im (z) − Im (p))2, which is simply the euclidean distance between z and p on the complex plane. So the magnitude of the transfer function at some point z essentially amounts to being the products of the distances between z and the system's zeros, divided by the products of the distances between z and the system's poles. The closer z is to a zero, the smaller the magnitude of the transfer function will be; the closer z is to a pole, the larger the magnitude of the transfer function will be.

Now recall that a system's frequency response, i.e., the amount the system scales the input at particular frequencies, is simply the z-transform evaluated on the unit circle, for $H(ej\omega)=H(z)|z=ej\omega$. If we have a pole-zero plot, we can quickly visualize the magnitude of the system's frequency response by traversing around the unit circle. As we go, the closer we are to a zero, the smaller it will be, while the closer we are to a pole, the larger it will be. As a result of this phenomenon (and the fact that the transfer function is determined by the pole and zero locations), we can see that the locations of a system's poles and zeros in the z-plane totally characterize a system (within a scaling factor). Thus, knowing the poles and zeros of a system is essentially equivalent to knowing exactly what kind of system it is.

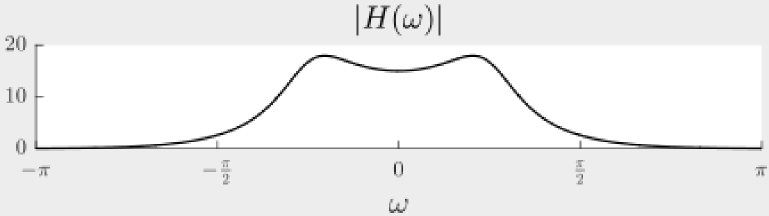
Visualizing the Magnitude of the Frequency Response

Consider the pole-zero plot from the example above, with a unit circle overlaid:



The frequency response of this system is simply value of the transfer function, evaluated along this unit circle. The magnitude of the transfer function is the product of the distances to the zeros, divided by the product of the distances to the poles. By traversing along the unit circle and noting these distances, we can visualize what the magnitude of the frequency response will be. We will start at $\omega=0$. Here we can easily find the value of the magnitude, since $H(ej0)=H(1)$. From the transfer function we found in the first example, we have: $H(1)=(1+3+114+34)(1-1+12)=15$.

So $|H(ej\omega)|$ at $\omega=0$ is 15. Now, as we move counter-clockwise around the circle (i.e., from $\omega=0$ to $\omega=\pi 2$ to $\omega=\pi$), note what happens. As ω approaches $\pi 4$ on the unit circle, it is drawing nearer and nearer to a pole; therefore, the magnitude of the frequency response is going to increase as ω progresses from 0 to $\pi 4$. At that point, we will be travelling away from that pole, and nearer to the three zeros; accordingly, the magnitude of the frequency response will decrease. Eventually we will actually land squarely on a zero at $\omega=\pi$; this is obviously as close as we could possibly get to a zero. As the distance to the zero at that point is zero, the magnitude of the frequency response there is zero. We can repeat this process moving clockwise from $\omega=0$ to $\omega=-\pi$ and the magnitude will change in the precisely same way, due to the symmetry. Having moved along the entire unit circle, we can see how our intuition matches what is indeed the magnitude of the frequency response, found via plotting software:



Note that the magnitude is 15 at $\omega=0$, then increases to a peak at about $\pm\pi 4$, and then decreases to 0 as ω traverses from $\pm\pi 4$ to $\pm\pi$.

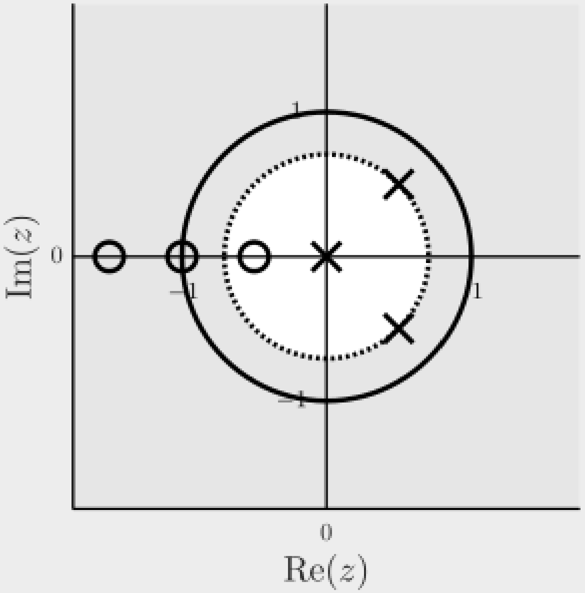
The Transfer Function ROC, and Stability

It has been explained previously that the region of convergence of a z-transform are the z for which the sum $\Sigma x[n]z-n$ converges. There is an alternative, though related definition that is also often used in signal processing, that the region of convergence are z for which the sum $\Sigma x[n]z-n$ converges *absolutely*, i.e. that $\Sigma|x[n]z-n|$. For virtually all of the signals that we will be considering, those two definitions are equivalent. The reason the second definition is often used is because of a relationship it establishes with the stability of the system. Suppose that a system's transfer function $H(z)$ has a particular ROC (defined in the absolutely summable sense). If the unit circle ($|z|=1$) is contained by this ROC, then the system is BIBO stable. To prove this fact, we see simply that the $|z|=1$ being in the ROC means that $\Sigma|h[n]ej\omega|$ converges for all ω, and then note that for $\omega=0$, the convergent sum is $\Sigma|h[n]|$. Previously we proved that for LTI systems, $\Sigma|h[n]|$ converging is equivalent to the system being BIBO stable.

So if the unit circle is in a transfer function's ROC, then that system is BIBO stable. This is a helpful way to determine stability. We do not need to prove that an arbitrary bounded input will produce a bounded output, nor find the impulse response and determine its absolute summability; we simply need to know the transfer function's ROC. If the system in question is causal (which is the case for most all practical systems of interest), then the ROC will extend outwards from the outermost pole. This means that, for causal systems, the location of the poles also determine BIBO stability. If all of a causal system's poles lie within the unit circle, then the ROC will extend to include the unit circle, and hence the system will be BIBO stable.

Stability and the Pole/Zero Plot

Consider again the pole/zero plot from the system in the previous examples:



Since the ROC contains the $|z| = 1$ unit circle, this system is BIBO stable.

Poles/Zeros and FIR/IIR Systems

We have seen that the poles and zeros of a system characterize it, which of course includes the nature of its frequency response and whether or not the system is BIBO stable. Given how significant the poles and zeros of a system are, it should come as no surprise that knowing about the poles and zeros also allows us to determine whether the system is FIR or IIR. Put simply, if a system's transfer function has any poles (other than at $z=0$), then the system is IIR. If it has only zeros, then it is FIR. This follows from what we saw with regard to the nature of different kinds of ROCs. If the ROC is the entire z plane (with the possible exception of $z=0$), then the impulse response is of finite duration; hence the system is FIR. But if there are any non-zero poles on the plane, than the ROC will either be a disk, an annulus, or will extend outside of a disk; in each of those cases, the impulse response is of infinite duration.

z-Transform Properties

As with other signal transforms, the z-transform has a number of significant properties, including ways in which changes to a signal in one domain impacts its representation in another domain. To examine these properties, we will again use the notation of a z-transform "pair." If we have $X(z)=\sum_{n=-\infty}^{\infty} x[n]z^{-n}$, then we express the relationship between $x[n]$ and its z-transform $X(z)$ as $x[n] \leftrightarrow X(z)$.

The z-Transform and the DTFT

The DTFT is a special case of the z-transform. If we have $x[n] \leftrightarrow X(z)$ and the ROC of $X(z)$ contains the unit circle, then the DTFT of $x[n]$, $X(e^{j\omega})$, is simply the z-transform $X(z)$ evaluated on the unit circle $z=e^{j\omega}$:

$$X(e^{j\omega}) = X(z)\big|_{z=e^{j\omega}}$$

The z-Transform is Linear

Given that the z-transform is simply an infinite sum, if follows then that it will be a linear operator. If we have $x_1[n] \leftrightarrow X_1(z)$ and $x_2[n] \leftrightarrow X_2(z)$, and $\alpha_1,\alpha_2 \in \mathbb{C}$, then:

$\alpha 1 x 1 [n] + \alpha 2 x 2 [n] \leftrightarrow \alpha 1 X 1 (z) + \alpha 2 X 2 (z)$

However, the ROC of the new transform is the intersection of those of X1(z) and X2(z):

$\mathrm{ROC}\alpha 1X1+\alpha 2X2=\mathrm{ROC}X1\bigcap \mathrm{ROC}X2.$

The z-Transform and Shifts in Time

If x[n] and X(z) are a z-transform pair, then: $x[n-m]\leftrightarrow z-mX(z)$ and the ROC remains the same as for X(z) (with the possible addition/removal of a pole at zero and/or zero at infinity):

$\Sigma n=-\infty \infty x[n-m]z-n=\Sigma r=-\infty \infty x[r]z-(r+m)=\Sigma r=-\infty \infty x[r]z-rz-m=z-m\Sigma r=-\infty \infty x[r]z-r=z-mX(z).$

The z-Transform and Modulation

If x[n] and X(z) are a z-transform pair, then: $z0nx[n]\leftrightarrow X(zz0)$. To determine the ROC of the new z-transform, substitute zz0 for z in the original ROC definition, and simplify. For example, if the ROC of X(z) is $|z|>1$, then the new ROC will be $|zz0|>1\rightarrow|z|>|z0|$:

Σn = − ∞ ∞ (z0nx[n])z − n = Σn = − ∞ ∞ x[n](z/z0) − n = X(zz0).

The z-Transform and Complex Conjugation

If x[n] and X(z) are a z-transform pair, then: x∗[n]⇔X∗(z∗):

Σn = − ∞ ∞ x∗[n]z − n = (Σn = − ∞ ∞ x[n](z∗) − n)∗ = X∗(z∗).

The z-Transform and Time Reversal

If x[n] and X(z) are a z-transform pair, then: x[− n]⇔X(z − 1). With the ROC inverted: to find the ROC, substitute 1z for z in the original ROC expression and simplify:

Σn = − ∞ ∞ x[− n]z − n = Σm = − ∞ ∞ x[m]zm = Σm = − ∞ ∞ x[m](z − 1) − m = X(z − 1).

The z-Transform and Convolution

If x[n]⇔X(z), h[n]⇔H(z), and y[n]⇔Y(z) all all z-transform pairs, and y[n] = x[n]∗h[n], then:

$Y(z)=X(z)H(z)$, with the ROC of $Y(z)$ being the intersection of the ROCs of $X(z)$ and $H(z)$.

Common z-Transform Pairs

In practice, one does not typically find the z-transform for a signal through the calculation of an infinite sum, but rather refers to a table to find a transform (along with the use of the properties listed above). Below is a table of some of the most common z-transform pairs:

x [n] X (z) ROC δ [n] 1 all   z u [n] 1 1 − z − 1 | z | > 1 u [− n − 1] 1 1 − z − 1 | z | < 1 α n u [n] 1 1 − α z − 1 | z | > | α | − α n u [− n − 1] 1 1 − α z − 1 | z | < | α |

The Inverse z-Transform

The z-transform for a signal x[n] has been defined as:

$$X(z)=\Sigma n=-\infty\infty x[n]z-n.$$

As with other transforms such as the DFT and DTFT, we would like to recover x[n] from its transform. The technical definition for this inverse transform is:

$$x[n]=\oint CX(z)zndzj2\pi z.$$

Evaluation of this contour integral requires knowledge of complex analysis. If you are handy with contour integrals, feel free to evaluate inverse transforms directly! But if you aren't, there is another way to find the inverse transform, which we will now consider.

Inverse Transform by Inspection

In order to find the inverse z-transform, we do not necessarily need to calculate anything; rather, we will seek to represent the z-transform in terms of entities for which we know the z-transform pair. For example, we know that the z-transform of $\alpha nu[n]$ is $11-\alpha z-1$, $|z|>\alpha$. So if we can represent some signal's z-transform in terms that look like $11-\alpha z$

-1, we will be on the right track to finding the inverse.

Key to this process is the fact that if a z-transform is rational (and it will be, for virtually all of our cases of interest), then its numerator and denominator can be factored according to its poles and zeros. This means that an X(z) of the form:

$$X(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_M z^{-M}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_N z^{-N}}$$

can equivalently be expressed as:

$$X(z) = z^{N-M}\frac{(1-\zeta_1 z^{-1})(1-\zeta_2 z^{-1})\cdots(1-\zeta_M z^{-1})}{(1-p_1 z^{-1})(1-p_2 z^{-1})\cdots(1-p_N z^{-1})}.$$

Note how this expression is starting to look similar to $\frac{1}{1-\alpha z^{-1}}$. We can do a little more work on X(z) to help with things even more. A rational function can be expressed in terms of products of $\frac{1}{1-\alpha z^{-1}}$ terms, but even better, it can also be represented as a *sum* of such terms. Therefore, any X(z) of the form

$$X(z) = z^{N-M}\frac{(1-\zeta_1 z^{-1})(1-\zeta_2 z^{-1})\cdots(1-\zeta_M z^{-1})}{(1-p_1 z^{-1})(1-p_2 z^{-1})\cdots(1-p_N z^{-1})}$$

can be re-arranged to be expressed as

$$X(z) = C_0 + \frac{C_1}{1-p_1 z^{-1}} + \frac{C_2}{1-p_2 z^{-1}} + \cdots$$

+CN1−pNz−1.

Now, this is something we can work with. Using the linearity property of the z-transform, we can take the inverse transform of that whole expression to yield (assuming the ROC of X(z) extends outwards from the outermost pole)
x[n]=C0δ[n]+C1p1nu[n]+C2p2nu[n]+⋯
+CNpNnu[n].
The plot of x [n] , the inverse z-transform of X (z) = 5 + 2 z − 1 − z − 2 1 − 1 6 z − 1 − 1 6 z − 2 , | z | > 1 2 .

Partial Fractions

Of course, we need to somehow find those C constants to be able to know the inverse transform. The process of finding X(z)=C0+C11−p1z −1+C21−p2z−1+⋯+CN1−pNz−1 from X(z)=b0+b1z−1+b2z−2+⋯+bMz −M1+a1z−1+a2z−2+⋯+aNz−N is known as a partial fraction decomposition.

The first step of the partial fraction decomposition is to make sure the order of the denominator polynomial is greater than that of the numerator. If not, then we first need to carry out a polynomial long division and then continue with the remainder. For example, suppose

X (z) = 5 + 2 z − 1 − z − 2 1 − 1 6 z − 1 − 1

6 z − 2 , | z | > 1 2

The long division is a single step:

6 1 − 1 6 z − 1 −
1 6 z − 2) 5 + 2 z − 1 − z − 2 ¯ 6 − 1 z
− 1 − z − 2 _ − 1 + 3 z − 1

So,

X(z)=5+2z−1−z−21−16z−1−16z−2=6+
−1+3z−11−16z−1−16z−2.

We can now work on that second term, as the denominator's order is greater than the numerator. The next step, then, will be to factor the denominator:

−1+3z−11−16z−1−16z−2= −1+3z
−1(1−12z−1)(1+13z−1).

From here, the next step is to express that in terms of a sum:

−1+3z−1(1−12z−1)(1+13z−1)=C11−12z
−1+C21+13z−1.The above form assumes that each of the poles is unique; if a pole is repeated, e.g. 1(1−az−1)2, then that single term in the product would correspond to two in the sum, C11−az
−1+C1′(1−az−1)2. At this point, we simply solve for the C coefficients:

−1+3z−1(1−12z−1)(1+13z−1)=C11−12z −1+C21+13z−1−1+3z−1=C1(1+13z −1)+C2(1−12z−1)−1+3z−1=(C1+C2)+z −1(13C1−12C2)C1+C2= −1, 13C1−12C2=3→C1=3 , C2=−4.

Given these coefficients, we now have that X(z)=6+31−12z−1−41+13z−1 , |z|>12.

Having split the transform into a sum through partial fractions, the inverse transform is now straightforward because (due to the linearity of the z-transform) we can take the inverse of each piece of the sum. So for:

X(z)=6+31−12z−1−41+13z−1 , |z|>12, the inverse transform of 1 is δ[n], the inverse of 11−12z−1 is (12)nu[n], and the inverse of 11+13z−1 is (−13)nu[n]. For the latter two terms, the ROC of the transform is must-know information; if the ROC extended inwards instead of outwards, the inverses would have been left-sided instead of right-sided. Putting all of this together through linearity, we have that the inverse transform is:

x[n]=6δ[n]+3(12)nu[n]−4(−13)nu[n].